

Lecture Notes in Artificial Intelligence 5972

Edited by R. Goebel, J. Siekmann, and W. Wahlster

Subseries of Lecture Notes in Computer Science

Norbert E. Fuchs (Ed.)

Controlled Natural Language

Workshop on Controlled Natural Language, CNL 2009
Marettimo Island, Italy, June 8-10, 2009
Revised Papers

Series Editors

Randy Goebel, University of Alberta, Edmonton, Canada

Jörg Siekmann, University of Saarland, Saarbrücken, Germany

Wolfgang Wahlster, DFKI and University of Saarland, Saarbrücken, Germany

Volume Editor

Norbert E. Fuchs

University of Zurich

Department of Informatics and Institute of Computational Linguistics,

Zurich, Switzerland

E-mail: fuchs@ifi.uzh.ch

Library of Congress Control Number: 2010930142

CR Subject Classification (1998): I.2.7, I.2, F.4.3, J.5, H.3-5, F.1

LNCS Sublibrary: SL 7 – Artificial Intelligence

ISSN 0302-9743

ISBN-10 3-642-14417-9 Springer Berlin Heidelberg New York

ISBN-13 978-3-642-14417-2 Springer Berlin Heidelberg New York

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable to prosecution under the German Copyright Law.

springer.com

© Springer-Verlag Berlin Heidelberg 2010

Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper 06/3180

Preface

Controlled natural languages (CNLs) are subsets of natural languages, obtained by restricting the grammar and vocabulary in order to reduce or eliminate ambiguity and complexity. Traditionally, controlled languages fall into two major types: those that improve readability for human readers, and those that enable reliable automatic semantic analysis of the language. [...] The second type of languages has a formal logical basis, i.e. they have a formal syntax and semantics, and can be mapped to an existing formal language, such as first-order logic. Thus, those languages can be used as knowledge representation languages, and writing of those languages is supported by fully automatic consistency and redundancy checks, query answering, etc.

Wikipedia

Various controlled natural languages of the second type have been developed by a number of organizations, and have been used in many different application domains, most recently within the Semantic Web.

The workshop CNL 2009 was dedicated to discussing the similarities and the differences of existing controlled natural languages of the second type, possible improvements to these languages, relations to other knowledge representation languages, tool support, existing and future applications, and further topics of interest. Specifically, CNL 2009 addressed the following aspects of controlled natural languages (CNLs):

- Design of CNLs
- Parsing of CNLs
- CNLs for knowledge representation
- CNLs for specifications
- CNLs and the Semantic Web
- CNLs as user interface
- CNLs for interaction and communication
- Tool support for CNLs
- Reasoning in CNLs
- Comparisons of CNLs
- Applications of CNLs
- Business rules
- User studies
- Theoretical results

The workshop was informal with lots of time for presentations and discussions in the fashion of the seminars organized at Dagstuhl in Germany. Based on the high number and the quality of the submissions, the large number of participants – altogether 36 researchers from 15 countries – and the positive feedback of the participants, the workshop can be considered a great success.

Researchers submitted 31 extended abstracts of which the Program Committee accepted 24. Two extended abstracts were withdrawn by their authors after acceptance.

Revised versions of the remaining 22 extended abstracts were published as *CEUR Workshop Proceedings (Volume 448)*.

During the workshop authors had ample time to present their work and to have it discussed by the participants. All authors of accepted extended abstracts were then invited to submit full papers taking the discussions during the workshop into account. Subsequently, 17 full papers were submitted of which the Program Committee accepted 16. This volume contains revised versions of these 16 full papers roughly divided into the two groups "Language Aspects" and "Tools and Applications." Note that some papers fall into both groups: using a controlled natural language in an application domain often requires domain-specific language features.

Additionally, this volume contains the invited paper "On Controlled Natural Languages: Properties and Prospects" by Adam Wyner et al. that summarizes a collaborative effort of the CNL community to define the concept "controlled natural language."

I would like to thank the authors of the extended abstracts and of the full papers for their contributions to the workshop. I also thank the members of the Program Committee and the additional reviewers for their great effort – first reviewing the extended abstracts and then the full papers – and for their constructive feedback that helped the authors to improve their papers. The submission and reviewing process and the compilation of the proceedings were greatly facilitated by the EasyChair system. Further thanks go to Randy Goebel, Joerg Siekmann and Wolfgang Wahlster – the editors of the series *Lecture Notes in Artificial Intelligence (LNCS/LNAI)* – and to the Springer staff for publishing the proceedings of CNL 2009. Last, but not least, I would like to thank my colleagues at the Department of Informatics and at the Institute of Computational Linguistics of the University of Zurich – Evelyn Berger, Kaarel Kaljurand, Tobias Kuhn, Cerstin Mahlow, and Michael Piotrowski – for generously helping me whenever I got stuck.

Il successo del seminario CNL 2009 non sarebbe stato possibile senza il generoso sostegno di Fausto Gobbo e del personale del Marettimo Residence, e senza il grande aiuto che ho ricevuto da Vito Vaccaro dell'Associazione Culturale, Sportiva, Ricreativa, Turistica "Marettimo". Vorrei cogliere questa occasione per ringraziarli tutti.

February 2010

Norbert E. Fuchs

Organization

The Workshop on Controlled Natural Language (CNL 2009) – organized by Norbert E. Fuchs of the University of Zurich – took place June 8-10, 2009 at the Marettimo Residence on the Sicilian island Marettimo. Further details can be found on the website of the workshop (<http://attempto.ifi.uzh.ch/site/cnl2009/>).

Program Committee

Piero Bonatti	University of Naples, Italy
Johan Bos	University of Rome "La Sapienza", Italy
Peter E. Clark	Boeing, Seattle, USA
Hamish Cunningham	University of Sheffield, UK
Enrico Franconi	University of Bolzano, Italy
Norbert E. Fuchs	University of Zurich, Switzerland (Chair)
Glen Hart	Ordnance Survey, Southampton, UK
Jerry R. Hobbs	USC/ISI, USA
Kaarel Kaljurand	University of Zurich, Switzerland
Peter Koepke	University of Bonn, Germany
Tobias Kuhn	University of Zurich, Switzerland
Ian Pratt-Hartmann	University of Manchester, UK
Stephen Pulman	University of Oxford, UK
Mike Rosner	University of Malta, Malta
Rolf Schwitter	Macquarie University, Australia
John Sowa	VivoMind, USA
Silvie Spreeuwenberg	LibRT, Amsterdam, The Netherlands
Uta Schwertel	imc, Germany
Yorick Wilks	University of Sheffield, UK

Additional Reviewers

William Murray	Boeing, Seattle, USA
----------------	----------------------

Local Organization

Norbert E. Fuchs	University of Zurich, Switzerland
Fausto Gobbo	Marettimo Residence, Italy
Vito Vaccaro	ACSRT Marettimo, Italy

Table of Contents

Language Aspects

An Evaluation Framework for Controlled Natural Languages	1
<i>Tobias Kuhn</i>	
Rhetorical Compositions for Controlled Natural Languages	21
<i>Andrew Potter</i>	
Anaphora Resolution Involving Interactive Knowledge Acquisition	36
<i>Rolf Schwitter</i>	
Talking Rabbit: A User Evaluation of Sentence Production	56
<i>Paula Engelbrecht, Glen Hart, and Catherine Dolbear</i>	
Naturalness vs. Predictability: A Key Debate in Controlled Languages	65
<i>Peter Clark, William R. Murray, Phil Harrison, and John Thompson</i>	
Implementing Controlled Languages in GF	82
<i>Krasimir Angelov and Aarne Ranta</i>	
Polysemy in Controlled Natural Language Texts	102
<i>Normunds Gruzitis and Guntis Barzdins</i>	
Economical Discourse Representation Theory	121
<i>Johan Bos</i>	
Controlled English Ontology-Based Data Access	135
<i>Camilo Thorne and Diego Calvanese</i>	
SBVR's Approach to Controlled Natural Language	155
<i>Silvie Spreeuwenberg and Keri Anderson Healy</i>	

Tools and Applications

The Naproche Project Controlled Natural Language Proof Checking of Mathematical Texts	170
<i>Marcos Cramer, Bernhard Fisseni, Peter Koepke, Daniel Kühlwein, Bernhard Schröder, and Jip Veldman</i>	
On Designing Controlled Natural Languages for Semantic Annotation	187
<i>Brian Davis, Pradeep Dantuluri, Laura Dragan, Siegfried Handschuh, and Hamish Cunningham</i>	

Development of a Controlled Natural Language Interface for Semantic MediaWiki	206
<i>Paul R. Smart, Jie Bao, Dave Braines, and Nigel R. Shadbolt</i>	
A Controlled Language for the Specification of Contracts	226
<i>Gordon J. Pace and Michael Rosner</i>	
Rabbit to OWL: Ontology Authoring with a CNL-Based Tool	246
<i>Ronald Denaux, Vania Dimitrova, Anthony G. Cohn, Catherine Dolbear, and Glen Hart</i>	
Writing Clinical Practice Guidelines in Controlled Natural Language ...	265
<i>Richard N. Shiffman, George Michel, Michael Krauthammer, Norbert E. Fuchs, Kaarel Kaljurand, and Tobias Kuhn</i>	

What Are Controlled Natural Languages?

On Controlled Natural Languages: Properties and Prospects	281
<i>Adam Wyner, Krasimir Angelov, Guntis Barzdins, Danica Damljanovic, Brian Davis, Norbert Fuchs, Stefan Hoefler, Ken Jones, Kaarel Kaljurand, Tobias Kuhn, Martin Luts, Jonathan Pool, Mike Rosner, Rolf Schwitter, and John Sowa</i>	
Author Index	291

An Evaluation Framework for Controlled Natural Languages^{*}

Tobias Kuhn

Department of Informatics & Institute of Computational Linguistics,
University of Zurich, Switzerland
tkuhn@ifi.uzh.ch
<http://www.ifi.uzh.ch/cl/tkuhn>

Abstract. This paper presents a general framework called *ontographs* that relies on a graphical notation and enables the tool-independent and reliable evaluation of human understandability of knowledge representation languages. An experiment with 64 participants is presented that applies this framework and compares a controlled natural language to a common formal language. The results show that the controlled natural language is easier to understand, needs less learning time, and is more accepted by its users.

1 Introduction

Controlled natural languages (CNL) have been proposed for the area of knowledge representation, and specifically for the Semantic Web, in order to overcome the problem that common formal languages are often hard to understand for people unfamiliar with formal notations [17,16]. User studies are the only way to verify whether CNLs are indeed easier to understand than other languages.

I propose here a novel approach for evaluating the understandability of CNLs. My approach relies on a graphical notation that I call *ontographs*. It allows for testing CNLs in a tool-independent way and for comparing them to other formal languages. The ontograph approach has been outlined in the extended abstract of this full paper [12]. Therein, the results of a first experiment are described. Here, I describe a second, more thorough experiment that has been conducted in the meantime. Both experiments and their results are described in more detail in my doctoral thesis [13].

Existing approaches to evaluate CNLs can be subdivided into two categories: task-based and paraphrase-based approaches. I will argue that it is difficult to get reliable results concerning the understandability of CNLs with either approach.

^{*} This work was funded by the research grant (Forschungskredit) programs 2006 and 2008 of the University of Zurich. I would like to thank Alain Cohn, Norbert E. Fuchs, Kaarel Kaljurand, Marc Lutz, Cerstin Mahlow, Philipp Müller and Michael Piotrowski for their suggestions and corrections. Furthermore, I would like to acknowledge the Institute for Empirical Research in Economics of the University of Zurich for organizing the recruitment of the participants for the experiment.

1.1 Task-Based CNL Experiments

In task-based experiments, the participants are given specific tasks to be accomplished by entering CNL statements into a given tool. One such experiment was described by Bernstein and Kaufmann [1], and another one was presented by Funk et al. [7,6]. In both cases, the participants of the experiment received tasks to add certain knowledge to the knowledge base using a tool that is based on a CNL.

An example taken from [7] is the task “Create a subclass *Journal of Periodical*” for which the participants are expected to write a CNL statement in the form of “Journals are a type of Periodicals”. To evaluate whether the task is accomplished, the resulting knowledge base can be checked whether it contains this actual piece of information or not. This approach bears some problems if used to test the understandability of a language.

First of all, such experiments mainly test the ability to write statements in the given CNL and not the ability to understand them. A user succeeding in the task shown above does not necessarily understand what the statement means. In order to add “Journal” as a subclass of “Periodical”, the user only has to map “subclass” and “type of”, but does not have to understand these terms.

Another problem is that it is hard to determine with such experiments how much the CNL contributes to the general usability and understandability, and how much is due to other aspects of the tool. It is also hard to compare CNLs to other formal languages with such studies, because different languages often require different tools. For these reasons, it would be desirable to be able to test the understandability of CNLs in a tool-independent way, which is not possible with task-based approaches. However, such approaches seem to be a good solution for testing the *writability* of CNLs.

1.2 Paraphrase-Based CNL Experiments

Paraphrase-based approaches are a way how CNLs can be tested in a tool-independent manner. In contrast to task-based approaches, they aim to evaluate the comprehensibility of a CNL rather than the usability of tools based on CNL.

Hart et al. [10] present such an approach to test their CNL (i.e. the Rabbit language). The authors conducted an experiment where the participants were given one Rabbit statement at a time and had to choose from four paraphrases in natural English, only one of which was correct. The authors give the following example of a Rabbit statement and four options:

Statement: Bob is an instance of an acornfly.

Option 1: Bob is a unique thing that is classified as an acornfly.

Option 2: Bob is sometimes an acornfly.

Option 3: All Bobs are types of acornflies.

Option 4: All acornflies are examples of Bob.

They used artificial words like “acornfly” in order to prevent that the participants classify the statements on the basis of their own background knowledge. Option 1

would be the correct solution in this case. Similar experiments are described by Hallett et al. [8] and Chervak et al. [4]. Again, there are some problems with such approaches.

First of all, since natural language is highly ambiguous, it has to be ensured somehow that the participants understand the natural language paraphrases in the way they are intended, which just takes the same problem to the next level. For the example above, one has to make sure that the participants understand phrases like “is classified as” and “are types of” in the correct way. The problem is even more complicated with words like “unique” and “sometimes”. If one cannot be sure that the participants understand the paraphrases then the results do not permit any conclusions about the understandability of the tested language.

Furthermore, since the formal statement and the paraphrases look very similar in many cases (both rely on English), it is yet again hard to determine whether understanding is actually necessary to fulfill the task. The participants might do the right thing without understanding the sentences (e.g. just by following some syntactic patterns), or by misunderstanding both — statement and paraphrase — in the same way. For the example above, a participant might just think that “an instance of” sounds like having the same meaning as “a unique thing that is classified as” without understanding any of the two. Such a person would be able to perform very well on the task above. In this case, the good performance would imply nothing about the understandability of the tested language.

Nevertheless, paraphrase-based approaches also have their advantages. One of them is that they scale very well with respect to the expressivity of the language to be tested. Basically, CNLs built upon any kind of formal logic can in principle be tested within such experiments, once the problems identified above are solved in one way or another.

2 The Ontograph Framework

In order to overcome the discussed problems of existing approaches, I propose a novel, diagram-based approach to test the understandability of languages. It relies on a graphical notation called *ontographs* (a contraction of “ontology graphs”). This notation is designed to be very simple and intuitive. The basic idea is to describe simple situations in this graphical notation so that these situation descriptions can be used in human subject experiments as a common basis to test the understandability of different formal languages. This approach allows for designing reliable understandability experiments that are completely tool-independent.

2.1 The Ontograph Notation

Every ontograph diagram consists of a legend that introduces types and relations and of a mini world that describes the actual individuals, their types, and their relations.

The legend of an ontograph introduces the names and the graphical representations of types and relations. Types are introduced by showing their name

beside the symbol that represents the respective type. For example, introducing a type “person” and another type “object” can be done as follows:



Starting from such general types, more specific ones can be defined. For example, “traveler” and “officer” can be defined as specific types of the general type “person”, and “present” and “TV” can be defined as specific types of “object”:

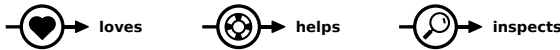


If a legend contains a type like “person” and, at the same time, a specific type like “traveler” then the part of the symbol of the specific type that is copied from the general type is displayed in gray:

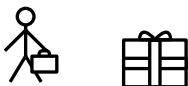


The purpose of this is to specify that only the black part of the symbol represents the respective type. This becomes important for individuals that belong to several types. For example, the suitcase is the deciding criterion in the case of the “traveler” definition (and not e.g. the missing hat).

Relations are represented by circles that contain a specific symbol with an arrow going through this circle. As with types, the legend introduces the names of the relations by showing them on the right hand side of their graphical representation. Some examples are the relations “loves”, “helps” and “inspects”:



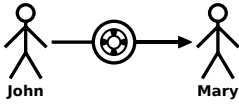
In contrast to the legend that only introduces vocabulary and the respective graphical representations, the mini world describes actual situations. Such situations consist of individuals, the types of the individuals, and the actual relations between them. Every individual is represented by exactly one symbol indicating the types of the individual. For example, an individual that is a traveler and another individual that is a present are represented by a traveler symbol and a present symbol:



An individual that belongs to more than one type is represented by a combined symbol that is obtained by merging the respective symbols. For example



represents an individual that is a traveler and an officer and another individual that is a present and a TV. Individuals can optionally have a name, which is shown in the mini world below the respective symbol. Relation instances are represented by arrows that point from one individual to another (or the same) individual and that have a relation symbol somewhere in the middle. “John helps Mary”, for example, would be represented as follows:



There is no explicit notation for negation. The fact that something is not the case is represented implicitly by not saying that it is the case. For example, stating that “John does not help Mary” is done by *not* drawing a *help*-relation from John to Mary. Thus, mini worlds are closed in the sense that everything that is true is shown and everything that is not shown is false.

Mini world and legend are compiled into a complete ontograph. The mini world is represented by a large square containing the mini world elements. The legend is represented by a smaller upright rectangle to the right of the mini world and contains the legend elements. Figure 1 shows an example.

2.2 Properties of the Ontograph Notation

The ontograph notation has some important characteristic properties. First of all, the ontograph notation does not allow for expressing incomplete knowledge. This means that nothing can be left unspecified and that every statement about the mini world is either necessarily true or necessarily false. For example, one can express “John helps Mary”, or one can also state “John does not help Mary”, but one cannot express that it is unknown whether one or the other is the case. Most other logic languages (e.g. standard first-order logic) do not behave this way. For the ontograph notation, this has the positive effect that no explicit notation for negation is needed.

Another important property of the ontograph notation is that it has no generalization capabilities. Logically speaking, the ontograph notation has no support for any kind of quantification over the individuals. For example, one cannot express something like “every man loves Mary” in a general way. The only way to express this is to draw a *love*-relation from every individual that is a man to the individual Mary. Thus, every individual and every relation instance has to be represented individually. This has the apparent consequence that the ontograph

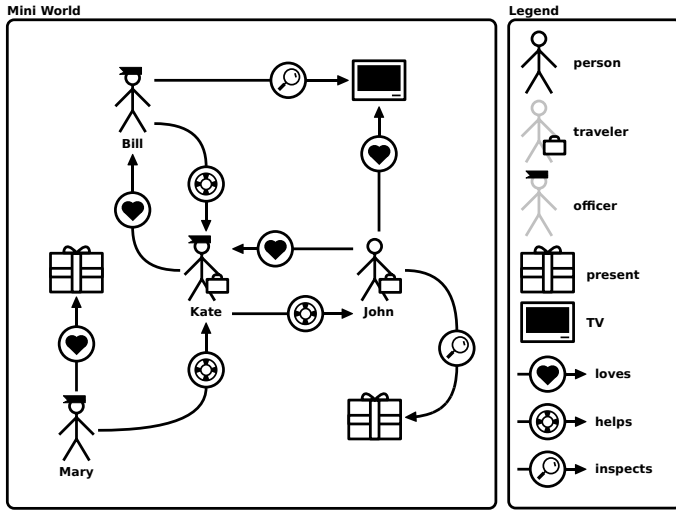


Fig. 1. This is an example of an ontograph. The legend on the right hand side defines the types and relations. The mini world on the left hand side shows the actual individuals, their types, and the relations between them.

notation cannot be used to describe situations with an infinite number of individuals and becomes impractical with something around 50 or more individuals and relation instances.

These properties make the ontograph notation a very simple language. They have also the consequence that the ontograph notation is no candidate for becoming a knowledge representation language of its own. A knowledge representation language without support for partial knowledge and generalization would not be very useful.

2.3 Ontograph Experiments

Ontographs are designed to be used in experiments testing the understandability of formal languages. They could, in principle, also be used to test the writability of languages by asking the participants to describe given situations. However, the latter approach has not yet been investigated.

In order to test the understandability of a language, an ontograph and several statements (written in the language to be tested) are shown to the participants of an experiment, who have to decide which of the statements are true and which are false with respect to the situation depicted by the ontograph.

Another important property of ontographs is that they use a graphical notation that is syntactically very different from textual languages like CNLs. This makes it virtually impossible to distinguish true and false statements of a given textual language with respect to a given ontograph just by looking at the syntax. If participants manage to systematically classify the statements correctly as

true or false then it can be concluded with a high degree of certainty that the participants understood the statements and the ontograph.

This point gets even clearer by applying a direct connection to model theory [3]. From a model-theoretic point of view, one can say that ontographs are a language to describe first-order models. The statements that are shown to the participants of an experiment would then be very simple first-order theories. From this point of view, the task of the participants is to decide whether certain theories have the shown ontograph as a model or not. We can say that participants understand a statement if they can correctly and systematically classify different ontographs as being a model of the statement or as not being a model thereof. This conclusion can be drawn because the truth of first-order theories can be defined solely on the basis of their models. Thus, being able to identify the ontographs that are models of a theory represented by a statement means that the respective person understands the statement correctly.

Admittedly, this model-theoretic interpretation of the term “understanding” is relatively narrow and ignores important problems like symbol grounding [9]. Such kinds of problems are not covered by the ontograph approach. Nevertheless, the ontograph framework allows us to draw stronger conclusions about the understandability of a language than other existing approaches.

2.4 Limitations and Related Approaches

The introduced ontograph approach, of course, also has its limitations. The most important one is probably the fact that only relatively simple forms of logic can be represented. Basically, ontographs cover first-order logic without functions and with the restriction to unary and binary predicates.

I do not see a simple solution at this point how predicates taking three arguments, for example, could be represented in an intuitive way. It would be even harder to represent more sophisticated forms of logic, e.g. modal or temporal logic. For such cases, it might be necessary to come back to task-based and paraphrase-based approaches to evaluate the understandability of languages. The core of such sophisticated forms of logic, however, could still be tested with the ontograph approach.

There are several existing approaches of using graphical notations to represent logical statements, for example Peirce’s *existential graphs* [14] and Sowa’s *conceptual graphs* [18]. However, such languages are fundamentally different from ontographs in the sense that they aim at representing general logical statements and in the sense that they are not designed to be intuitively understandable but have to be learned.

The combination of intuitive pictures and statements in natural language can also be found in books for language learners, e.g. “English through pictures” [15]. As in the ontograph framework, pictures are used as a language that is understood without explanation.

The idea of “textual model checking” presented by Bos [2] is similar to the ontograph approach in some respects. Like in the ontograph approach, there is the task of classifying given statements as true or false with respect to a

given situation. In contrast to the approach presented here, the task is to be performed by computer programs and not by humans, and it is about testing these computer programs rather than the language.

3 Experiment Design

The presented ontograph framework has been applied to test whether Attempto Controlled English (ACE) [5], which is a controlled subset of English, is easier to understand than a comparable common formal language. The experiment to be described was performed on 64 participants. Further design decisions are described below in more detail.

3.1 Comparable Language

The most important design decision for the experiment is the choice of the language to which ACE is compared. For this experiment, the Manchester OWL Syntax, a usability-oriented syntax of the ontology language OWL, has been chosen. The inventors of the Manchester OWL Syntax introduce it as follows [11]:

The syntax, which is known as the Manchester OWL Syntax, was developed in response to a demand from a wide range of users, who do not have a Description Logic background, for a “less logician like” syntax. [...] This means that it is quick and easy to read and write.

As this quotation shows, the Manchester OWL Syntax is designed for good usability and good understandability and thus seems to be an appropriate choice for this experiment. However, the Manchester OWL Syntax requires the statements to be grouped by their main ontological entity (the one in subject position so to speak). This is a reasonable approach for the definition of complete ontologies, but it makes it impossible to state short and independent statements that could be used for a direct comparison to ACE in an experimental setting. For this reason, a modified version of the Manchester OWL Syntax has been defined specifically for this experiment. The resulting language, which I will call “Manchester-like language” or “MLL”, uses the same or very similar keywords but allows us to state short and independent statements.

3.2 Learning Time

Obviously, the understanding of a language highly depends on the amount of time spent for learning the language. This means that one has to define a certain time frame when evaluating the understandability of languages. Some languages might be the best choice if there is only little learning time; other languages might be less understandable in this situation but are more suitable in the long run.

So far, little is known about how the understandability of CNLs compares to the understandability of common formal languages. CNLs are designed to be

understandable with no learning and the results of the first ontograph experiment [12] show that this is indeed the case. Since other formal languages like the Manchester OWL Syntax are not designed to be understandable with no learning at all, it would not be appropriate to compare ACE to such a language in a zero learning time scenario.

For this reason, I chose a learning time of about 20 minutes. This seems to be a reasonable first step away from the zero learning time scenario. The effect of longer learning times remains open to be studied in the future.

3.3 Ontographs and Statements

Four series of ontographs have been created that cover certain types of statements: The first series contains only individuals and types without relations; the statements of the second series contain relations with different kinds of simple universal quantifications; the third series contains domain, range, and number restrictions; the fourth series, finally, consists basically only of relations.

For each of the four series, three ontographs have been created. For each ontograph, 20 statement pairs have been defined in a way that each pair consists of an ACE statement and a semantically equivalent MLL statement. Some of the statement pairs are true with respect to their ontograph and the others are false.

Table 1 shows examples of statements in their representations in ACE and MLL. It also shows how the statements are divided into four series. All statements together with their ontograph diagrams are available in my doctoral thesis [13] and online¹.

3.4 Participants

Another important design decision is the choice of the participants. Such studies are mostly performed with students because they are flexible and usually close to the research facilities of the universities. In my case, there are even more reasons why students are a good choice. Students are used to think systematically and logically but they are usually not familiar with formal logical notations (unless this lies in their field of study). In this way, they resemble domain experts who have to formalize their knowledge and who should profit from languages like ACE.

The requirements for the participants have been defined as follows: They had to be students or graduates with no higher education in computer science or logic. Furthermore, at least intermediate level skills in written German and English were required, because the experiment itself was explained and performed in German, and English was needed to understand the ACE sentences.

64 students have been recruited who fulfill these requirements and exhibit a broad variety of fields of study. The students were on average 22 years old and 42% of them were female and 58% were male.

¹ <http://attempto.ifi.uzh.ch/site/docs/ontograph/>

Table 1. This table shows examples of statements in their representations in ACE and MLL. These statements are divided into four series.

Series ACE	MLL
1	
Mary is a traveler.	Mary HasType traveler
Bill is not a golfer.	Bill HasType not golfer
Mary is an officer or is a golfer.	Mary HasType officer or golfer
Sue is an officer and is a traveler.	Sue HasType officer and traveler
Every man is a golfer.	man SubTypeOf golfer
No golfer is a woman.	golfer DisjointWith woman
Every woman is an officer and every officer is a woman.	woman EquivalentTo officer
Every traveler who is not a woman is a golfer.	traveler and (not woman) SubTypeOf golfer
Every man is a golfer or is a traveler.	man SubTypeOf golfer or traveler
Nobody who is a man or who is a golfer is an officer and is a traveler.	man or golfer SubTypeOf not (officer and traveler)
2	
Lisa sees Mary.	Lisa sees Mary
Mary does not see Tom.	Mary not sees Tom
Tom buys a picture.	Tom HasType buys some picture
Mary sees no man.	Mary HasType not (sees some man)
John buys something that is not a present.	John HasType buys some (not present)
John sees nothing but men.	John HasType sees only man
Every man buys a present.	man SubTypeOf buys some present
Everything that buys a present is a man.	buys some present SubTypeOf man
Every man buys nothing but presents.	man SubTypeOf buys only present
Everything that buys nothing but pictures is a woman.	buys only picture SubTypeOf woman
3	
Everything that inspects something is an officer	inspects HasDomain officer
Everything that is inspected by something is a letter.	inspects HasRange letter
Everything that inspects something is a golfer or is an officer.	inspects HasDomain golfer or officer
Everything that is seen by something is an officer or is a picture.	sees HasRange officer or picture
Lisa inspects at least 2 letters.	Lisa HasType inspects min 2 letter
Lisa helps at most 1 person.	Lisa HasType helps max 1 person
Every officer helps at least 2 persons.	officer SubTypeOf helps min 2 person
Everything that sees at least 2 pictures is an officer.	sees min 2 picture SubTypeOf officer
Every person inspects at most 1 letter.	person SubTypeOf inspects max 1 letter
Everything that is an officer or that is a golfer sees at most 1 picture.	officer or golfer SubTypeOf sees max 1 picture
4	
If X helps Y then Y helps X.	helps IsSymmetric
If X sees Y then Y does not see X.	sees IsAsymmetric
If X sees somebody who sees Y then X sees Y.	sees IsTransitive
If X admires Y then X sees Y.	admires SubRelationOf sees
If X inspects Y then X helps Y.	inspects SubRelationOf helps
If X helps Y then Y admires X.	helps SubRelationOf inverse admires
If X loves Y then X does not admire Y.	loves DisjointWith admires
If X sees Y then Y does not love X.	sees DisjointWith inverse love
If X admires Y then X sees Y. If X sees Y then X admires Y.	admires EquivalentTo sees
If X inspects Y then Y sees X. If Y sees X then X inspects Y.	inspects EquivalentTo inverse sees

In order to enable a good comparison between the two languages, each participant was tested on ACE and on MLL. However, since participants cannot be expected to concentrate for much longer than one hour, only one of the four ontograph series could be tested per participant.

In order to rule out learning effects, half of the participants received the ACE task first and then the MLL task while the other half received the tasks in the reverse way.

3.5 Procedure

The experiment was conducted in a computer room with a computer for each participant. The main part of the experiment was performed on the computer screen. Additionally, the participants received different printed sheets during the experiment. The overall procedure consisted of six stages:

1. Instructions with Control Questions
2. Learning Phase 1
3. Testing Phase 1
4. Learning Phase 2
5. Testing Phase 2
6. Questionnaire

For the instruction phase, the participants received a printed instruction sheet that explained the experiment procedure, the payout², and the ontograph notation. The reverse side of the instruction sheet contained control questions for the participants to answer, which allowed us to check whether the participants understood the instructions correctly. The participants had to return the filled-out instruction sheets to the experimenter, who checked whether all questions were answered correctly. In the case of false answers, the experimenter explained the respective issue to the participant.

For the first learning phase, the participants received a language description sheet of the first language (either ACE or MLL). This language description sheet only explained the subset of the language that is used for the respective series. For this reason, each series had its own instruction sheets for both languages. During the learning phase, the participants had to read the language description sheet. Furthermore, an ontograph (the same as on the instruction sheet) was shown on the screen together with 10 true statements marked as “true” and 10 false statements marked as “false” in the respective language. Figure 2 shows a screenshot of the experiment screen during the learning phase.

During the testing phase, a different ontograph was shown on the screen. Furthermore, 10 statements in the respective language were shown on the screen together with radio buttons that allowed the participants to choose between “true”, “false” and “don’t know”. Figure 3 shows how the experiment screen of the testing phase looked like. During the testing phase, the participants could

² Apart from a fixed fee, the participants received an additional small amount of money for each correctly classified statement and half of it for “don’t know” answers.

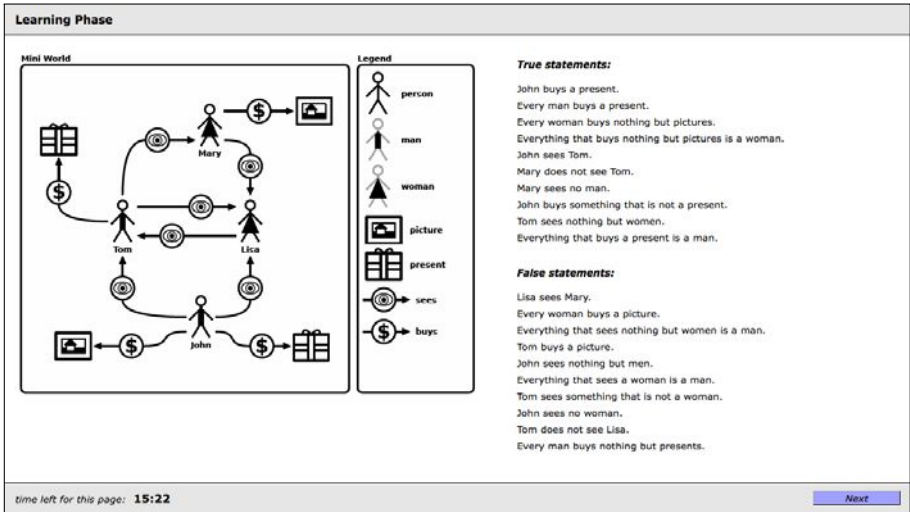


Fig. 2. This is the screen shown during the learning phase of the experiment

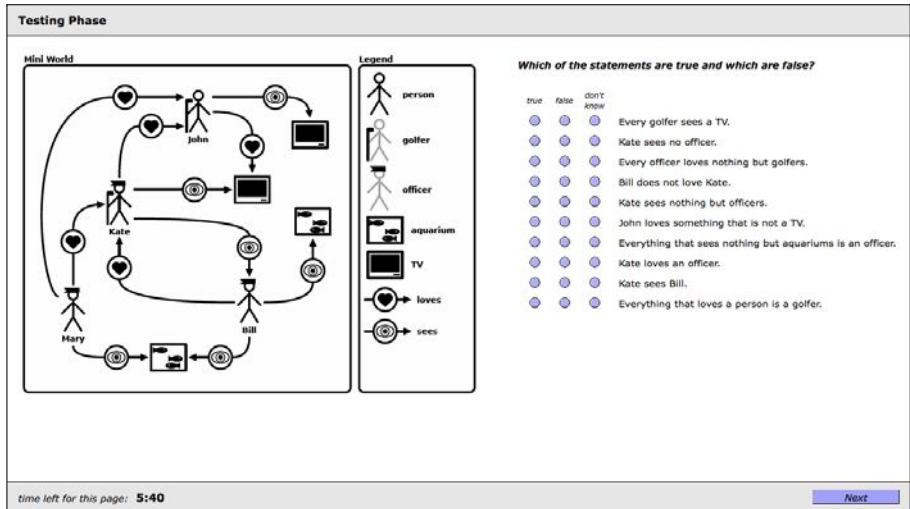


Fig. 3. This is the screen shown during the testing phase of the experiment

keep the language description sheet that they got for the learning phase. Thus, they did not need to know the language description by heart but they could read parts of it again during the testing phase if necessary.

For the steps 4 and 5, the procedure of the steps 2 and 3 was repeated for the second language (i.e. ACE if the first language was MLL and vice versa) with the same ontograph for the learning phase but a new one for the testing.

Finally, the participants received a questionnaire form inquiring about their background and their experiences during the experiment. The experiment was finished when the participants turned in the completed questionnaire form.

The learning phases had a time limit of 16 minutes each, and the time limit for the testing phases was 6 minutes. The participants were forced to proceed when the time limit ran out but they could proceed earlier. In this way, it can not only be investigated how understandable the languages are but also how much time the participants needed to learn them.

3.6 Language Description Sheets

The proper design of the language description sheets is crucial for this experiment. If the participants perform better in one language than in the other, it might be that the respective language was merely described better than the other. Thus, the language description sheets have to be written very carefully to be sure that they are not misunderstood and are optimal for learning the respective language under the given time restrictions. Especially the description sheets for MLL are critical. In contrast to ACE, MLL is not designed to be understood without training. For this reason, a special effort has been made to ensure the quality of the MLL description sheets. This quality assurance effort involved several steps.

First of all, the four series were designed in a way that at most seven MLL keywords are used per series. Since each series has its own language description sheets, not more than seven keywords have to be described by the same sheet. This should make it easier to understand the needed subset of the language.

In a second step, the different MLL description sheets were given to three persons, who comply with the restrictions of the experiment but who did not participate in it. These three persons read the sheets and gave me feedback about what they did not understand and what could be improved.

As a third step, I performed a test run with 9 participants to receive final feedback about the understandability and usefulness of the language description sheets. After the test run, the participants received the sheets again and they were told to highlight everything that was difficult to understand. Only very few things were highlighted (altogether two highlightings in the MLL description, one in the ACE description, and none in the general instructions) and according to this I made a couple of small last changes for the main experiment.

Altogether, the language description sheets were compiled very carefully and it is very unlikely that a different description of MLL would radically increase its understandability.

4 Results

The results of the experiment allow for comparing ACE and MLL on the basis of the classification results, the time required by the participants, and the answers they gave in the questionnaire. It also has to be evaluated whether the ontograph framework altogether worked out or not.

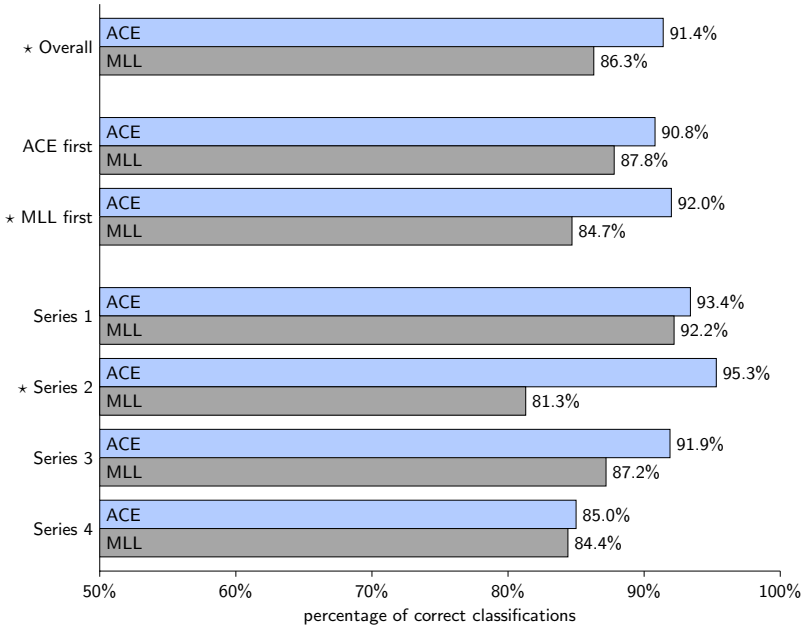


Fig. 4. This chart shows the percentage of correct classifications. The base line is 50% that can be achieved by mere guessing. “Don’t know”-classifications and cases where the time limit ran out count as 0.5 correct classifications. Significant differences are marked by “*” (see Table 2 for details).

4.1 General Classification Scores

Figure 4 shows the average percentages of correct classifications per testing phase. “Don’t know” answers and the cases where the time limit ran out are counted as 0.5 correct classifications. 50% is the baseline because an average of five correct classifications out of ten can be achieved by mere guessing (or, for that matter, by choosing always “don’t know” or by letting the time limit run out).

91.4% of the statements were classified correctly in the case of ACE and 86.3% in the case of MLL. Thus, out of the ten statements of a testing phase, ACE was on average 0.5 points better than MLL. This is a considerable and statistically significant difference (the details of the used statistical test to compare the two samples are explained later on). One has to consider that these values are already close to the ceiling in the form of the perfect score of 10, which might have reduced the actual effect.

The results of the participants who received ACE first and then MLL can now be compared with the ones who received MLL first. As expected, both languages were understood better when they were the second language. This can be explained by the fact that the participants were more familiar with the procedure, the task, and the ontograph notation. However, even in the case when

ACE was the first language and MLL the second one, ACE was understood better (but in this case not within statistical significance).

Looking at the results from the perspective of the different series, one can see that ACE was better in all cases but only the series 2 and 3 exhibit a clear dominance of ACE (and this dominance is significant only for series 2). According to these results, one could say that languages like MLL are equally easy to understand for very simple statements as the ones in series 1 and for statements about relations as they appear in series 4. In the case of series 1, the reason might be that these statements are so simple that they can be understood even in a rather complicated language. In the case of series 4, the reason is probably that Description Logic based languages like MLL can express these statements without variables whereas ACE needs variables, which are somehow borderline cases in terms of naturalness.

In summary, the results show that — while both languages are understood reasonably well — ACE is easier to understand than MLL.

4.2 Time

As a next step, we can look at the time values. For simplicity reasons and since the learning process was presumably not restricted to the learning phase but continued during the testing phase, the time needed for both phases will together be called the *learning time*.

Figure 5 shows the learning times of the participants. They could spend at most 22 minutes: 16 minutes for the learning phase and 6 minutes for the testing phase. The results show that the participants needed much less time for ACE than for MLL. In the case of ACE less than 14 minutes were needed, whereas in the case of MLL the participants needed more than 18 minutes. Thus, MLL required 29% more time to be learned, compared to ACE.

Note that these results can be evaluated only together with the results described above concerning the classification scores. The learning time can only be evaluated together with the degree of understanding it entails. The smaller amount of learning time for ACE can be explained simply by the fact that the language description sheets for ACE contained less text than the ones for MLL. But together with the results described above that show that ACE was understood better and the fact that the language description sheets have been written very carefully, it can be concluded that ACE required less learning time while leading to a higher degree of understanding.

Again, we can split the results according to the participants who received ACE first and those who received MLL first. The results show the expected effect: ACE and MLL required less time as second language. However, ACE required less time than MLL no matter if it was the first language or the second. Thus, even in the cases where ACE was the first language and the participants had no previous experience with the procedure and MLL was the second language and the participants could use the experiences they made before, even in such cases ACE required less time.

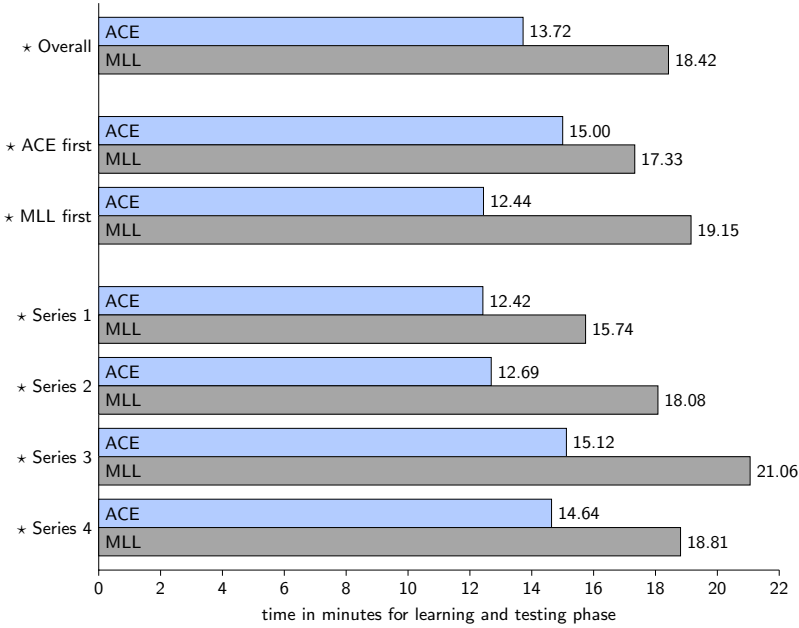


Fig. 5. This chart shows the average time needed for learning and testing phase. Significant differences are marked by “★” (see Table 2 for details).

Looking at the different series, we can see that this effect spreads over all four series. MLL required on average between 3 and 6 minutes more learning time than ACE.

The better time values of ACE compared to MLL are statistically significant for the whole sample and also for all presented subsamples.

4.3 Perceived Understandability

As a third dimension, we can look at the “perceived understandability”, i.e. how the participants perceived the understandability of the languages. The questionnaire that the participants filled out after the experiment contained two questions that asked the participants how understandable they found ACE and MLL, respectively. They could choose from four options: “very hard to understand” (value 0), “hard to understand” (1), “easy to understand” (2) and “very easy to understand” (3). The perceived understandability does not necessarily have to coincide with the actual understandability and can be a very valuable measure for the acceptance of a language and the confidence of its users.

Figure 6 shows the scores for perceived understandability derived from the questionnaire. Overall, ACE got much better scores than MLL. MLL was close but below “easy to understand” scoring 1.92, whereas ACE was closer to “very easy to understand” than to “easy to understand” scoring 2.59.

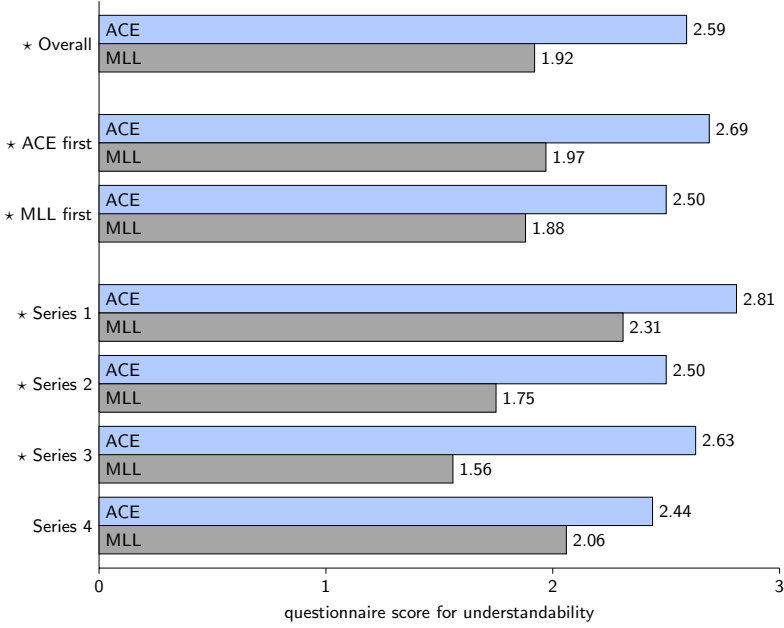


Fig. 6. This chart shows the average scores for perceived understandability derived from the questionnaire. 0 means “very hard to understand”, 1 means “hard to understand”, 2 means “easy to understand”, and 3 means “very easy to understand”. Significant differences are marked by “★” (see Table 2 for details).

By dividing the results into those who received ACE first and those who received MLL first, we see that both languages scored better when ACE was the first language. I do not have a convincing explanation for this and it might just be a statistical artifact.

Looking at the perceived understandability scores from the perspective of the different series, we can see that ACE clearly received better scores in all four series. It is interesting that this also holds for the series 1 and 4 where ACE was not much better than MLL in terms of actual understanding, as shown before. Thus, even though the actual understanding of the statements of these series does not show a clear difference, the acceptance and confidence of the participants seems to be higher in the case of ACE.

4.4 Significance

The charts with the experiment results indicate in which cases the difference between ACE and MLL is statistically significant. This was done by using the Wilcoxon signed-rank test [19], which is a non-parametric statistical method for testing the difference between measurements of a paired sample. In contrast to Student’s *t*-test, this test does not rely on the assumption that the statistical

Table 2. This table shows the p -values of Wilcoxon signed-rank tests. The null hypothesis is that the given values are not different for ACE and for MLL. This null hypothesis can be rejected in 16 of the 21 cases on a 95% confidence level and these cases are marked by “★”.

	classification score		time		questionnaire score	
complete sample	0.003421	★	1.493×10^{-10}	★	3.240×10^{-7}	★
ACE first	0.2140		0.006640	★	7.343×10^{-5}	★
MLL first	0.005893	★	3.260×10^{-9}	★	0.001850	★
Series 1	0.5859		0.01309	★	0.02148	★
Series 2	0.003052	★	0.002624	★	0.02197	★
Series 3	0.1250		9.155×10^{-5}	★	0.0004883	★
Series 4	0.6335		0.002686	★	0.1855	

population corresponds to a standard normal distribution. This relieves us from investigating whether standard normal distribution can be assumed for the given situation.

Table 2 shows the obtained p -values for the three dimensions of our comparison (i.e. classification score, time, and questionnaire score). For the complete sample, the values are well within the 95% confidence level for all three dimensions. They are even within the 99% level.

4.5 Framework Evaluation

Finally, it can be evaluated whether the ontograph framework altogether worked out or not.

Figure 7 shows the results of two questions of the questionnaire asking the participants about how understandable they found the ontograph notation and the overall instructions. Both values are between “easy to understand” and “very easy to understand”. This shows that the ontographs were well accepted by the participants and that it is possible to explain the procedure of such experiments in an understandable way.

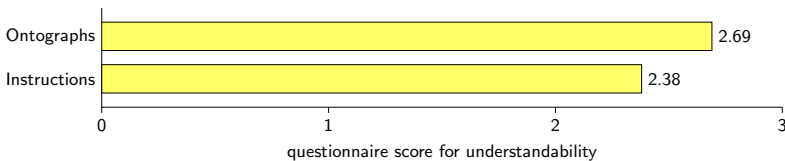


Fig. 7. This chart shows the average understandability scores for the ontograph notation and the instructions, derived from the questionnaire. 0 means “very hard to understand”, 1 means “hard to understand”, 2 means “easy to understand”, and 3 means “very easy to understand”.

Furthermore, the results of the experiment show that the ontographs were indeed very well understood by the participants. For both languages, the overall percentage of correct classifications exceeded 85%. Such good results are only possible if the ontographs and the instructions are understood.

5 Conclusions

The results of the two experiments show that the ontograph framework worked out very well and is suitable for testing the understandability of languages. I could show that ACE is understood significantly better than the comparable language MLL. Furthermore, ACE required much less time to be learned and was perceived as more understandable by the participants.

MLL is directly derived from the Manchester OWL Syntax in a way that leaves its properties concerning understandability intact. For this reason, the conclusions of the experiment can be directly applied to the Manchester OWL Syntax, which is the state of the art approach on how to represent ontological knowledge in a user-friendly manner. Thus, it could be shown that CNLs like ACE can do better in terms of understandability than the current state of the art.

Altogether, the results suggest that CNLs should be used instead of languages like the Manchester OWL Syntax in situations where people have to deal with knowledge representations after little or no training.

References

1. Bernstein, A., Kaufmann, E.: GINO — a guided input natural language ontology editor. In: Cruz, I., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., Aroyo, L.M. (eds.) ISWC 2006. LNCS, vol. 4273, pp. 144–157. Springer, Heidelberg (2006)
2. Bos, J.: Let's not Argue about Semantics. In: Proceedings of the Sixth International Language Resources and Evaluation (LREC 2008), European Language Resources Association (ELRA), pp. 2835–2840 (2008)
3. Chang, C.C., Keisler, H.J.: Model Theory. Studies in Logic and the Foundations of Mathematics, vol. 73. North-Holland, Amsterdam (1973)
4. Chervak, S., Drury, C.G., Ouellette, J.P.: Field evaluation of simplified english for aircraft workcards. In: Proceedings of the 10th FAA/AAM Meeting on Human Factors in Aviation Maintenance and Inspection (1996)
5. Fuchs, N.E., Kaljurand, K., Kuhn, T.: Attempto Controlled English for knowledge representation. In: Baroglio, C., Bonatti, P.A., Małuszyński, J., Marchiori, M., Polleres, A., Schaffert, S. (eds.) Reasoning Web. LNCS, vol. 5224, pp. 104–124. Springer, Heidelberg (2008)
6. Funk, A., Davis, B., Tablan, V., Bontcheva, K., Cunningham, H.: Controlled language IE components version 2. SEKT Project Deliverable D2.2.2, University of Sheffield, UK (2007)
7. Funk, A., Tablan, V., Bontcheva, K., Cunningham, H., Davis, B., Handschuh, S.: CLOnE: Controlled language for ontology editing. In: Aberer, K., Choi, K.-S., Noy, N., Allemang, D., Lee, K.-I., Nixon, L.J.B., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudré-Mauroux, P. (eds.) ASWC 2007 and ISWC 2007. LNCS, vol. 4825, pp. 142–155. Springer, Heidelberg (2007)

8. Hallett, C., Scott, D., Power, R.: Composing questions through conceptual authoring. *Computational Linguistics* 33(1), 105–133 (2007)
9. Harnad, S.: The symbol grounding problem. *Physica D: Nonlinear Phenomena* 42(1-3), 335–346 (1990)
10. Hart, G., Johnson, M., Dolbear, C.: Rabbit: Developing a controlled natural language for authoring ontologies. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) *ESWC 2008. LNCS*, vol. 5021, pp. 348–360. Springer, Heidelberg (2008)
11. Horridge, M., Drummond, N., Goodwin, J., Rector, A.L., Stevens, R., Wang, H.: The Manchester OWL syntax. In: *Proceedings of the OWLED 2006 Workshop on OWL: Experiences and Directions. CEUR Workshop Proceedings*, vol. 216, CEUR-WS (2006)
12. Kuhn, T.: How to evaluate controlled natural languages. In: *Pre-Proceedings of the Workshop on Controlled Natural Language (CNL 2009)*, CEUR Workshop Proceedings, April 2009, vol. 448. CEUR-WS (2009)
13. Kuhn, T.: *Controlled English for Knowledge Representation*. Doctoral thesis, Faculty of Economics, Business Administration and Information Technology of the University of Zurich, Switzerland, (to appear)
14. Peirce, C.S.: Existential graphs. In: *Collected Papers of Charles Sanders Peirce. The Simplest Mathematics*, vol. 4. Harvard University Press, Cambridge (1932)
15. Richards, I.A., Gibson, C.M.: *English through Pictures*. Washington Square Press (1945)
16. Schwitter, R., Kaljurand, K., Cregan, A., Dolbear, C., Hart, G.: A comparison of three controlled natural languages for OWL 1.1. In: *Proceedings of the Fourth OWLED Workshop on OWL: Experiences and Directions, CEUR Workshop Proceedings*, vol. 496. CEUR-WS (2008)
17. Schwitter, R., Tilbrook, M.: Controlled Natural Language meets the Semantic Web. In: *Proceedings of the Australasian Language Technology Workshop 2004, ALTA Electronic Proceedings*, vol. 2, pp. 55–62. Australasian Language Technology Association (2004)
18. Sowa, J.F.: *Knowledge Representation: Logical, Philosophical, and Computational Foundations*. Brooks/Cole (2000)
19. Wilcoxon, F.: Individual comparisons by ranking methods. *Biometrics Bulletin* 1(6), 80–83 (1945)

Rhetorical Compositions for Controlled Natural Languages

Andrew Potter

Sentar, Incorporated, 315 Wynn Drive, Suite 1
Huntsville, Alabama, USA 35805
andrew.potter@sentar.com

Abstract. Logic-based controlled natural languages usually provide some facility for compositional representation, minimally including sentence level coordination and sometimes subordination. These forms of compositional representation are useful for expressing short passages of a few sentences, but compared to natural language they offer only a limited range of expression, and they are unwieldy for expressing longer passages. This paper describes a method for bringing more naturalness to controlled natural languages. This approach defines a model for representing compositional structures found in natural language, such as Antithesis, Concession, and Unless in a way that is both rhetorically expressive and logically reducible. This is demonstrated through a series of examples using both natural and controlled natural language. The paper then describes a set of intercompositional rules of inference that can be used to reason about the interrelationships between compositional structures, facilitating the discovery and assessment of supportive and conflicting relationships among them. This ability to represent and discover interrelationships among discourse representation structures could be useful for developing applications that must not only follow a set of rules of inference, but reason about the rules themselves.

Keywords: Controlled Natural Language, Rhetorical Structure Theory, Argumentation, Knowledge Representation, Logic, Explanation Aware Computing.

1 Introduction

Logic-based controlled natural languages usually provide some facility for compositional representation. Most well known among these, ACE and PENG define discourse representation structures that support sentence level coordination and subordination [1, 2], and CLCE, CPL, and E2V support sentence level coordination [3-5]. These forms of compositional representation are useful for expressing short passages of a few sentences, but relative to natural language they offer only a limited range of expression, and they soon become unwieldy for expressing longer passages. New techniques are needed for representing compositions in a way that is both rhetorically expressive and logically reducible.

In this paper we describe an approach to addressing this need. We first describe the theoretical foundations for our approach, based on Rhetorical Structure Theory and

the Toulmin model of argumentation. We then show that complex compositional relations found in natural language can be reduced to propositional logic, and we show how these relations can be applied to controlled natural languages to produce more expressive texts. Because the structures are reducible to logical expressions, it is possible to use them to construct highly expressive knowledge representations. Next we introduce a set of intercompositional rules of inference that can be used to reason about the chaining interrelationships among these compositional structures, facilitating the discovery and evaluation of supportive and conflicting relationships among them. Finally, we conclude with a discussion of the significance of these observations, as well as some directions for future research.

2 Theoretical Basis

The approach presented here defines a mapping between Rhetorical Structure Theory [6] and the Toulmin model of argumentation [7]. RST supplies rhetorical relations and structural constraints used to define expressive compositional representations in a logically reducible manner. The Toulmin model provides an argumentative framework for applying a set of rhetorical rules of inference. Just as the Toulmin model uses a warrant to establish a link between a ground and a claim, RST establishes links between a nucleus and its satellites. An example of this mapping is shown in Fig. 1, where two statements are related using the RST Evidence relation, such that the satellite corresponds to the argumentative ground and the nucleus corresponds to the claim. By mapping Toulmin with RST, we can say that an RST relation is used to characterize a warrant in terms of its specific relationship between its claim and its ground, and moreover, that any claim may have any number of grounds, just as in RST a nucleus may have multiple satellites.

More formally, we can say that this model defines warrants, spans, statements, relations, and inference rules, such that a *warrant* establishes a set of links between a nucleus and zero or more satellites, as shown in Fig. 2. The nucleus and its satellites are represented as *spans*. A span consists of a CNL *statement* and, in the case of satellites, the satellite's *relation* to its nucleus. In argumentative terms, the nucleus corresponds to a claim, and the satellites correspond to grounds. Each satellite (or ground) links to the nucleus (or claim) by means of a rhetorical relation. Note that a statement can be either a elementary discourse unit or it can be another warrant. This permits the construction of nested structures.

For the purposes of this analysis, RST relations are treated as either *inferential* or *synthetic*. An inferential relation is one whose satellite has an argumentative, causal, or logical implication for the nucleus, such that if the satellite is instantiated, the nucleus may be inferred. Examples of inferential relations include Antithesis, Condition, Concession, Evidence, Unless, and the various causal relations defined by RST. In contrast to inferential relations, synthetic relations are purely informational, so that the satellite tends to provide information relating to the nucleus, but the satellite is not material to the nucleus. Examples of synthetic relations are Background, Circumstance, Elaboration, and Summary. *Intercompositional* inference rules are defined for characterizing the interrelationships among argumentative structures; for example, one instance of Evidence will *substantiate* another if its claim (nucleus) unifies with

the ground (satellite) of the other. These intercompositional rules of inference and their significance for knowledge representation will be explored in detail later in this paper.

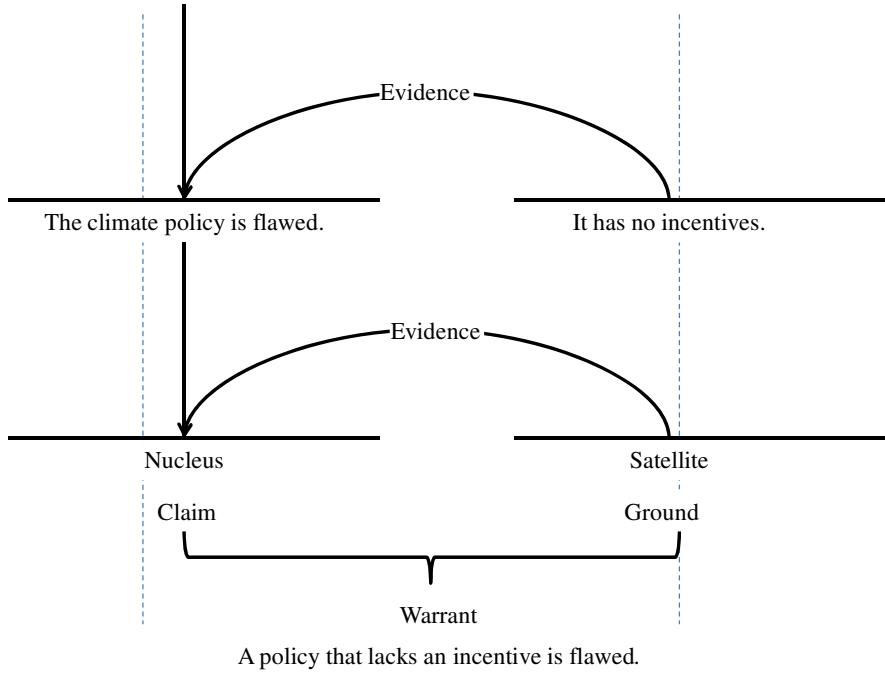


Fig. 1. Toulmin RST mapping shows the correspondence between an inferential rhetorical relation and a warrant

3 The Logic of Rhetorical Relations

Having outlined the underlying theoretical considerations, we can make this more concrete by showing through examples how rhetorical relations can be mapped into propositional logic. In the following, we give examples of key inferential rhetorical relations using natural language. Once we have established that this approach is applicable to natural language, we can then take the next step, of applying the same techniques to logic-based controlled natural languages to produce more expressive compositional structures.

3.1 Antithesis and Concession

Antithesis and *Concession* are closely related rhetorical relations. *Antithesis* is used to increase the reader's positive regard for the nucleus by introducing incompatible information in the satellite that casts the nucleus in a more positive light. Similar to *Antithesis*, the *Concession* relation seeks to increase the reader's positive regard for

the nucleus. However, it accomplishes this not by introducing incompatible information in the satellite, but by acknowledging an *apparent* incompatibility between the nucleus and satellite in such a way that the effect is that the reader’s positive regard for the nucleus is increased. So the difference between the two is subtle. Indeed, Stede [8] argued that the difference is so subtle that perhaps there is not one. He claimed that Antithesis and Concession are nearly identical, except that Antithesis is more general since, unlike Concession, it places no constraints in the satellite. However, there is another important difference—namely that in the Antithesis relation there is an incompatibility between the satellite and nucleus, whereas with Concession the incompatibility is merely apparent.

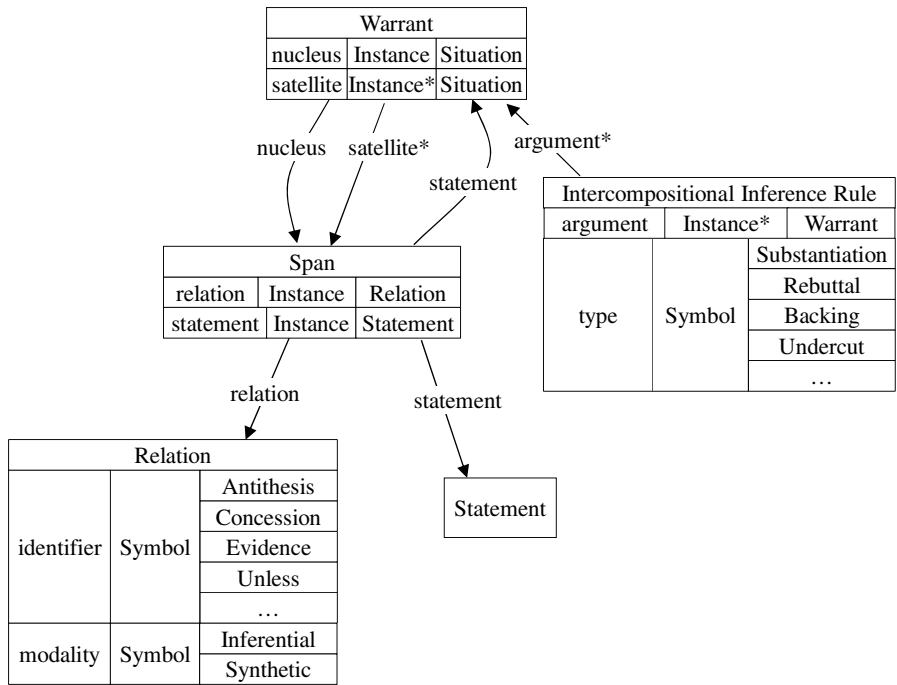


Fig. 2. The Reasoning Model defines a mapping between RST and the Toulmin model

An analysis of the underlying logic of Antithesis and Concession suggests that while the given definitions of these two relations are similar, there are important differences in logical structure. This can be seen clearly through examination of a few examples:

- 1) What engenders respect is not the particular outcome that a legal scholar arrives at but, rather, the intellectual rigor and honesty with which he or she arrives at a decision.

—Barack Obama

This example consists of two segments, and they are clearly intended as incompatible. And clearly the writer (or speaker) has a preference for the latter of the two. Hence the relation is one of Antithesis. Now if the rhetorical structure can be mapped to an argument structure, then the argument ground and claim should correspond to the satellite and nucleus, and we need to supply a plausible warrant for binding the two. The *claim* is that respect is engendered by the intellectual rigor and honesty with which a legal scholar arrives at a decision; the *ground* is that respect is not engendered by the particular outcome the scholar arrives at. So how does the claim follow from the ground? Here is a logical interpretation of Antithesis:

$$(\sim(g \equiv c) \cdot \sim g) \supset c \quad (1)$$

That is, the ground *g* and claim *C* are incompatible, and since not *g*, *C*. That this is a plausible interpretation can be seen by revisiting the original text. Having set up the incompatibility between these two means of engendering respect, we negate one of them, hence implying the other. To this extent we establish positive regard for the nucleus by means of indicating negative regard for the satellite. However, we cannot expect that every instance of Antithesis will include a satellite stated with explicit negative quality:

- 2) I'd rather lose an election than see my country lose a war.

—John McCain

While it might seem unlikely that any politician would under any circumstances prefer to lose an election, it is this expectation that gives the statement its rhetorical punch. The speaker identifies two incompatible situations, and makes clear, given the choice, his preference between the two. Though there is no explicit negation in the satellite, the speaker's positive regard for the nucleus is clearly indicated. Thompson and Mann [9] found that in natural language (English) the use of the contrastive conjunction *Rather* often signals Antithesis.

Concession increases positive regard for the nucleus by acknowledging an apparent incompatibility, but by recognizing that this incompatibility is only apparent, the writer increases reader's positive regard for the situation presented in nucleus. If the satellite and nucleus were really incompatible, then the nucleus might be cast in doubt:

- 3) And I thought we could muddle through without the governor's personal chef
— although I've got to admit that sometimes my kids sure miss her.

—Sarah Palin

There is no incompatibility between muddling through without the chef and admitting that the kids sure missed her. That is, since *g* does not imply that *C* is false, *C* must be true:

$$\sim(g \supset \sim c) \supset c \quad (2)$$

As a generalized structure, such patterns cannot be escalated to prescriptive argument schema, because not all instantiations will be true [10]. But as a persuasive pattern of reasoning, Concession occurs frequently in natural language, and therefore

is a structure of interest for controlled natural languages. The argument trades on the lack of incompatibility between the satellite and nucleus. This is quite different from Antithesis, where the argument trades on the recognition of their incompatibility.

3.2 Evidence and Other Modus Ponens Relations

In the *Evidence* relation, comprehension of the satellite increases the reader's belief of the Nucleus. That is, the satellite provides evidence for the nucleus. Here is an example:

- 4) There is far too much arresting of people going on. Not just Skip [Henry Louis] Gates — a couple of weeks ago, some poor woman in Montauk was arrested in her bathing suit, marched off the beach and charged with a felony for forging a beach parking sticker.

—Nora Ephron

The claim is that “far too much arresting of people going on,” and in support of this claim, the writer cites a particular example as evidence. The logical structure of the Evidence relation, as well as many other inferential relations, is *modus ponens*.

$$((g \supset c) \cdot g) \supset c \quad (3)$$

Other inferential relations following this pattern are *Condition*, *Justify*, *Motivation*, *Enablement*, *Means*, and *Purpose*, as well as the causal relations, *Nonvolitional-Cause*, *Nonvolitional-Result*, *Volitional-Cause*, and *Volitional-Result*. These relations all follow the familiar modus ponens logical structure. Note that with Motivation and Enablement, the nuclear claim is not an assertion of fact, but an action. Here is an example of Motivation:

- 5) We do, quite literally, possess the means to destroy all of mankind. We must seek to do all we can to ensure that nuclear weapons will never again be used.

—John McCain

In this example the claim is a call to action motivated by the threat posed by the current availability of nuclear weapons. Just as with Evidence, where the reader might not believe the nucleus without the satellite, but with Motivation, the satellite increases the reader's desire to perform action presented in nucleus.

3.3 Unless

In Rhetorical Structure Theory, the definition of *Unless* states that the Nucleus is realized provided that the Satellite is not. In argumentative terms, this states that negation of the ground implies the claim:

$$\sim g \supset c \quad (4)$$

One might reasonably take a step further here and say that the RST definition also entails that if the ground is not negated, then the claim is negated:

$$g \supset \sim c \quad (5)$$

Or, in other words

$$\sim(g \equiv c) \quad (6)$$

which corresponds to what has been referred to in logic as the *strong sense* of Unless [11]. And clearly the connective is sometimes used in this way:

- 6) Unless it's a life or death emergency, no credible doctor is going perform surgery on a child for any reason without a parent's permission.

—Richard Shelby

Surgery on a child without the parent's permission will be performed if and only if there is life or death emergency. But in the *weak sense*, which is the more common use of Unless, a negative ground implies the claim, but a positive ground does not necessarily negate the ground. Here is an example:

- 7) And unless we free ourselves from a dependence on these fossil fuels and chart a new course on energy in this country, we are condemning future generations to global catastrophe.

—Barack Obama

Given that the options for global catastrophe are numerous, charting a new course on energy will not guarantee a future free of global catastrophe; it will eliminate one possible catastrophic scenario. The presence of the cue word “unless” makes an RST encoding of the relation Unless difficult to resist, even when the weak sense is intended.

If the Unless relation is to be used in a CNL, it will be necessary to pick one interpretation and stick with it. Since the weak sense is the more common usage in natural language, it seems reasonable that this would be the preferred interpretation. But there is another reason for adopting the weak sense of Unless. Unless can be combined with other arguments to serve as a Toulmin qualifier. In the Toulmin model, a qualifier is used to indicate that the linkage between the ground and the claim may be subject to exception. That is, the Unless relation limits the warrant, as in this example:

- 8) When the Federal Reserve extends a loan or purchases a security, this automatically adds reserves to the banking system unless the Fed undertakes an offsetting reserve draining operation.

—William C. Dudley

In this example, the warrant is that activities such as extending loans put money into the system. The draining operation is cited as a circumstance when the warrant would not apply. So in the absence of the exception the warrant holds:

$$\sim g \supset w \quad (7)$$

But it seems unreasonable to suggest that the negative ground and the warrant are materially equivalent, as would be required by the strong sense. The difficulty becomes apparent when we expand the warrant,

$$\sim g_q \supset (g_w \supset c_w) \quad (8)$$

since $(g_w > c_w)$ will hold anytime c_w , regardless of whether g_w . In our example, the strong sense would require that a draining operation had been undertaken any time no addition to the banking systems was made. The use of Unless as a qualifier brings us to a further observation. We need not be limited to relations between elementary discourse units; indeed we can construct relations between units and structures to create more complex structures, and we can even relate one structure to another. Further, these structures need not necessarily be known during composition, but may be discovered and integrated during runtime reasoning. Used in this way, Unless is an example of a particular intercompositional rule of inference call dissociation, to be discussed in detail later in this paper.

4 CNL and the Logic of Rhetorical Relations

We have now seen how inferential rhetorical relations can be mapped into propositional logic. This indicates that the resulting compositions are not “merely rhetorical,” but that they are also shorthand for complex logical expressions. To the extent that this mapping is possible in natural language, it may also be a desirable feature for controlled natural languages. To explore this, we will construct examples using simple ACE statements as elements and add rhetorical relations for sentence level coordination. We can then compare these compositions with logically equivalent ACE compositional representations. The rhetorical relations used here are Concession, Antithesis, Causality, and Unless.

4.1 Concession

As discussed earlier, Concession appeals to an apparent incompatibility between ground and claim, such that the claim is strengthened by acknowledgement of the ground. That is, it is not the case that the ground denies the claim, and therefore the claim is seen as more likely. To illustrate this in CNL, we start with two statements:

- 9) Some individuals believe that the climate problem is a scientific hoax.
- 10) The climate problem is not a scientific hoax.

and develop a Concession relation between the two, signaled by the connective *although*:

- 11) Although some individuals believe that the climate problem is a scientific hoax, the climate problem is not a scientific hoax.

Without the Concession relation, the CNL equivalent of this would be:

- 12) If it is false that if some individuals believe that the climate problem is a scientific hoax then it is false that the climate problem is not a scientific hoax then the climate problem is not a scientific hoax.

That is, that some individuals believe that the climate problem is a scientific hoax does not mean that it is a hoax, and that lends support to the claim that it is not. Although the reasoning underlying the Concession relation may be *argumentum ad ignorantium*, support for compositional representations employing *although* would nevertheless give writers a useful tool for making statements that are currently impracticable with the current CNL technology.

4.2 Antithesis

Antithesis appeals to an incompatibility between ground and claim, but unlike Concession, the incompatibility is claimed to be real rather than apparent:

- 13) The climate problem is a scientific hoax.
- 14) The climate is a critical problem.

An antithetical relationship can be established when we negate the first statement and apply *rather* as a connective between the two:

- 15) The climate problem is not a scientific hoax, rather the climate is a critical problem.

Using the logical definition given earlier (1), without support for antithetical compositions, the example can be written in CNL as:

- 16) If it is false that if the climate problem is a scientific hoax then the climate is a critical problem, and it is false that if the climate is a critical problem then the climate problem is a scientific hoax, and it is false that the climate problem is a scientific hoax then the climate is a critical problem.

Making this readable requires more than fixing up the commas. The statements 15) and 16) are logical equivalent, but the negated material equivalence embedded in Antithesis is more readily expressed using *rather* rather than nested conditionals. While natural language affords numerous ways to construct antithetical expressions [9], the ability to use *rather* for sentence level coordination would clearly lend power to any CNL.

4.3 Causality

Rhetorical structure theory defines a variety of volitional and non-volitional causal relations, but for our purposes we need only one composite causal relation, which we may call *Because*. Using the causal relation we can construct the following compositional relation from a set of simple statements:

- 17) All scientists know that the climate is a problem, because the atmosphere is polluted, and the sea-levels are higher, and the average temperatures are higher.

which conveys a sense of causality not expressible without the Because relation. At best, we can create compound conditional, such as the following:

- 18) If the atmosphere is polluted, and the sea-levels are higher, and the average temperatures are higher then all scientists know that the climate is a problem.

Although the gain in readability achieved through use of the causal relation is less than that of relations such as Concession and Antithesis, by identifying the causal nature of the relationship between ground and claim, we do provide richer expressiveness than is possible relying on the if-then structure.

4.4 Unless

As noted earlier, the Unless relation, as used in natural language, indicates that the claim of an argument is realized provided that the ground is not. Here is an example using two CNL statements:

- 19) Unless all emissions are controlled soon, the climate problem is hopeless.

with the logical equivalent of

- 20) If all emissions are not controlled soon then the climate problem is hopeless.

The weak form of Unless is used here, as is preferable for CNL, in order to support the its use as a Toulmin qualifier, as discussed earlier. The qualifier indicates that the linkage between the argument ground and claim may be subject to exception. The weak form allows this to happen, while having no effect on the argument when there is no exception. We can construct a CNL example of Unless as a Toulmin qualifier from the following elements:

- 21) No rich countries lead.
- 22) Some developing countries make no improvements.
- 23) The developing countries experience a global-warming disaster.

First we construct the argument:

- 24) If no rich countries lead then some developing countries make no improvements.

And then we qualify it:

- 25) If no rich countries lead then some developing countries make no improvements, unless they experience a global-warming disaster.

According to this argument, some developing countries require the leadership of rich countries in order to undertake self-improvement on their own, but if things get bad

enough, they may initiate these improvements on their own. Here is how this can be represented in CNL without the Unless relation:

- 26) If some developing countries experience a global-warming disaster then it is false that if no rich countries lead then the developing countries make no improvements.

As with the other examples we have seen here, while the statements are logically equivalent, the naturalness of the content is overshadowed by the need for explicit logical controls. As will be developed in the discussion of intercompositional rules of inference, the availability of the Unless relation would provide a powerful tool for specifying the terms of dissociation within an argument.

5 Intercompositional Rules of Inference

Thus far we have focused on how the logic of rhetorical relations could be used to write more expressive compositional representations in CNL. However, taking the view of these compositions as warrants suggests some interesting possibilities—namely that we might be able to define rules that enable automated reasoning about the structures, thus identifying the interrelationships among them. These rules, called intercompositional rules of inference, are identified in in Table 1. Basically, these rules identify the kinds of inferential chaining that may occur among compositions. An intercompositional inference occurs when the nucleus, satellite, or warrant of one structure can be unified with the nucleus, satellite, or warrant of some other structure. In the *Substantiation* rule, the claim of one argument is used as the ground of another; this contrasts with *Undercut*, where the claim of one argument is incompatible with the ground of another. With *Concomitance*, two arguments share the same ground to establish distinct claims, whereas with *Confusion*, the grounds of two arguments are incompatible. With *Rebuttal*, the claims of two arguments are incompatible, while with *Convergence*, two grounds converge upon a shared claim. The *Backing* rule states that the claim of one argument substantiates the warrant of another, whereas with *Dissociation*, the claim of one argument disputes the warrant of another. In the following discussion, we take a closer look at some of these rules.

Table 1. Intercompositional Rules of Inference

<i>Rule</i>	<i>Definition</i>
Substantiation	The claim of one argument is used as the ground of another
Rebuttal	The claims of two arguments are incompatible
Backing	An argument substantiates the warrant of another
Undercut	The claim of one argument is incompatible with the ground of another
Dissociation	The claim of one argument disputes the warrant of another
Convergence	Two arguments lead to the same claim
Concomitance	Two arguments use the same ground to establish distinct claims
Confusion	The grounds of two arguments are incompatible

5.1 Substantiation

Substantiation occurs when the claim of one argument unifies with the ground of another. Substantiating arguments may be chained to one another through a claim-ground linkage. Here is a straightforward example using simple conditional arguments:

- 27) If the climate problem is not a scientific hoax then the climate is a critical problem.
- 28) If the climate problem is a critical problem, then it is necessary that all governments act soon.

This seems simple enough. However, since the Substantiation rule applies whenever the claim of one argument unifies with the ground of another, it is possible to use other relations, such as Concession and Antithesis, to allow for more complex inferences based on the Substantiation rule. In the following example, the ground that some individuals believe that the climate problem is a scientific hoax ultimately substantiates claim that the climate is a critical problem:

- 29) Although some individuals believe that the climate problem is a scientific hoax, the climate problem is not a scientific hoax.
- 30) The climate problem is not a scientific hoax, rather the climate is a critical problem.

5.2 Backing

Backing is similar to Substantiation, except that the claim being substantiated is a warrant, consisting of both ground and claim. That is, an argumentative relation is substantiated, not just its ground or its claim. Here is an example of a Backing rule:

- 31) It is true that if no rich countries lead then some developing countries experience a global-warming disaster, because they lack the necessary motivation.

In this argument, the claim consists of the conditional statement it is true that if no rich countries lead some developing countries experience a global-warming disaster, and the ground is because some developing countries lack the necessary resources. That is, the strength of the conditional is grounded by its backing. Using structures such as this, a knowledge base can achieve added explanatory power. In a fully developed knowledge base, an instance of the Backing rule could be the claim of some further argumentation.

5.3 Dissociation

If we wish to have the means to explain why an argument should be accepted, then we should also provide for when it should not. Thus the Backing rule is counter-balanced by *Dissociation*. We have already seen some examples of Dissociation in the discussion of the Unless relation as a Toulmin qualifier. The effect of Unless is to *dissociate* the ground from claim, thereby limiting the warrant, for example:

- 32) If no rich countries lead then some developing countries make no improvements, unless they experience a global-warming disaster.

By the Dissociation rule, the warrant that the absence of improvement among developing countries could be attributable to a lack of leadership from richer countries would be inapplicable when circumstances included a global-warming disaster. In other words, any instantiation of the rule

- 33) If no rich countries lead then some developing countries make no improvements.

is weakened by the qualifier, if the qualifier is also instantiated. Thus the Unless relation makes it possible to make this determination at runtime. The use of the Unless relation maps easily to the dissociation rule, as it is the classic Toulmin exception. However, other relations can be used to implement Dissociation. Here is an example using Antithesis:

- 34) It is not true that if the climate problem is a critical problem, then it is necessary that all governments act soon, rather nobody has a solution.

The Concession relation can be used similarly:

- 35) Although it is not true that if the climate problem is a critical problem, then it is necessary that all governments act soon, nobody has a solution.

Thus there are several ways to dissociate a warrant. The Unless relation limits a warrant by identifying possible exceptions to it, Concession weakens it by suggesting that it might not be to the point, and Antithesis rejects it outright.

5.4 Convergence

Convergence is a more difficult case. The Convergence rule applies when multiple arguments lead to the same claim. In natural language, Convergence is difficult to analyze logically because its structure supports multiple semantics. As noted by Walton [12], in a purely convergent argument, multiple premises support a conclusion, but they do so independently. Yet there are also arguments of identical structure where the premises combine to support the conclusion. These Walton refers to as *linked* arguments. In a linked argument, neither premise alone is sufficient to establish the conclusion. In contrast, a merely convergent argument is actually multiple arguments which share the same conclusion. In natural discourse, it is not always easy to distinguish linked from convergent arguments. Consider this example,

- 36) All scientists know that the climate is a problem, because the atmosphere is polluted, and the sea-levels are higher, and the average temperatures are higher.

In this argument, the conclusion is supported by several conditions. But how should this be interpreted? Would scientists still know that the climate is a problem if

one or two of the three conditions were not met? It might be the case that any one of these conditions would be sufficient to reach the conclusion. So it is possible that this example is a case of argumentative accrual, or what Walton calls an *evidence-accumulating* argument [12].

But if there are evidence-accumulating arguments, then there might also be evidence-dissipating arguments, where the arguments in a convergence cancel one another out—as in the example used by Prakken [13], where one argument uses rain as a reason not to go jogging and the argument other uses heat as a reason not to go jogging, but with the result that the two arguments converge upon a single claim. It is important to note that this interpretation relies on ontological insight—one must understand something about jogging in order to detect the cancellation effect.

So there are four possible interpretations of a convergence structure: the argument could be convergent, it could be linked, it could be evidence-accumulating, or it could be evidence dissipating. For CNL, it would be possible to signal whether a structure is linked, accumulating, dissipating, or merely convergent through the use of cue words. For example, a term such as furthermore could be used to indicate an evidence-accumulating argument. However, this approach does not fully address the problem, because convergent structures may also be discovered at runtime [14]. For example, suppose we have a knowledge base containing the following arguments:

- 37) All scientists know that the climate is a problem, because the atmosphere is polluted.
- 38) All scientists know that the climate is a problem, because the sea-levels are higher.
- 39) All scientists know that the climate is a problem, because the average temperatures are higher.

Using the Convergence rule, we can infer a convergent structure. Since the statements occur independently, we may also infer that they are not linked. But there is no structural basis for supposing that the structure is evidence-accumulating or evidence-dissipating. Thus, the minimal case, that convergent structures discovered a posteriori are simple convergences seems reasonable—in other words, the rule at work here is a *convergence* rule, and not a *linked-argument*, *evidence-accumulating*, *evidence-dissipating* rule. The most that can be inferred at runtime is a convergence relationship.

6 Conclusion

In this paper, we have examined composite discourse structures in natural language and used this to develop new possibilities for expressiveness in controlled natural languages. By showing that rhetorical relations are reducible to propositional logic, we suggest that they could be incorporated into CNLs to provide writers with enhanced resources for expressiveness while assuring that any resulting expressions could be parsed and interpreted by a reasoning system. Enabling CNL writers to incorporate such composite expressions into their knowledge representations would enable them to achieve added explanatory power.

Intercompositional rules of inference open the possibility of extending the scope of compositional activity beyond authorship to include runtime discovery of supportive and conflicting relationships among compositional structures. This ability to represent and discover interrelationships within and among structures in an explicit way could prove useful for developing applications that not only follow a set of rules of inference, but reason about the rules themselves. This suggests the possibility of introducing techniques for defeasible reasoning into controlled natural languages. For example the current approach could be extended to include the use of qualifiers and qualification ratios [15] for use in representing argument defeasibility, with resulting qualification values propagating through networks of rhetorical structures. It also seems possible that some rhetorical relations may be construed as stronger than others; for example, arguments based on causality may be stronger than arguments based on Concession. Additional research is needed to understand the full ramifications and utility of these possibilities.

References

1. Fuchs, N.E., Kaljurand, K., Kuhn, T.: Discourse representation structures for ACE 6.0. Department of Informatics. University of Zurich, Zurich (2008)
2. Schwitter, R.: English as a formal specification language. In: *Proceedings of the 13th International Workshop on Database and Expert Systems Applications*, pp. 228–232 (2002)
3. Pratt-Hartmann, I.: A two-variable fragment of English. *Journal of Logic, Language and Information* 12, 13–45 (2003)
4. Sowa, J.: *Common logic controlled English* (2007)
5. Clark, P., Harrison, P., Jenkins, T., Thompson, J., Wojcik, R.: Acquiring and using world knowledge using a restricted subset of English. In: *FLAIRS*, pp. 506–511 (2005)
6. Mann, W.C., Thompson, S.A.: Rhetorical structure theory: Towards a functional theory of text organization. *Text* 8, 243–281 (1988)
7. Toulmin, S.E.: *The uses of argument*. Cambridge University Press, Cambridge (1958)
8. Stede, M.: Disambiguating rhetorical structure. *Research on Language & Computation* 6, 311–332 (2006)
9. Thompson, S.A., Mann, W.C.: Antithesis: A study in clause combining and discourse structure. In: Steele, R., Threadgold, T. (eds.) *Language Topics: Essays in Honour of Michael Halliday*, vol. II, pp. 359–381. John Benjamins, Amsterdam (1987)
10. Walton, D.: Explanations and arguments based on practical reasoning. In: *Proceedings of Workshop W10: Explanation-Aware Computing, Twenty-First International Joint Conference on Artificial Intelligence*, Pasadena, pp. 72–83 (2009)
11. Hardegree, G.: *Symbolic logic: A first course*. McGraw-Hill, New York (1999)
12. Walton, D.N.: *Argument structure: A pragmatic theory*. University of Toronto Press, Toronto (1996)
13. Prakken, H.: A Study of Accrual of Arguments, with Applications to evidential reasoning. In: *The Tenth International Conference on Artificial Intelligence and Law, Proceedings of the Conference, June 6–11*, pp. 85–94. ACM, New York (2005)
14. Potter, A.: Linked and convergent structures in discourse-based reasoning. In: Roth-Berghofer, T., Schulz, S., Bahls, D., Leake, D.B. (eds.) *Proceedings of the 3rd International Explanation Aware Computing Workshop (ExaCt 2008)*, Patras, Greece, pp. 72–83 (2008)
15. Fox, J., Das, S.: *Artificial intelligence in hazardous applications*. AAAI Press, Menlo Park (2000)

Anaphora Resolution Involving Interactive Knowledge Acquisition

Rolf Schwitter

Centre for Language Technology
Macquarie University
Sydney 2109 NSW, Australia
Rolf.Schwitter@mq.edu.au

Abstract. Anaphora resolution in current computer-processable controlled natural languages relies mainly on syntactic information, accessibility constraints and the distance of the anaphoric expression to its antecedent. This design decision has the advantage that a text can be processed automatically without any additional ontological knowledge, but it has the disadvantage that the author is severely restricted in using anaphoric expressions while writing a text. I will argue that we can allow for a wider range of anaphoric expressions whose resolution requires inference-supporting knowledge, if we consider the anaphora resolution process as an interactive knowledge acquisition process in those cases where no suitable noun phrase antecedent can be found. In particular, I will focus on definite descriptions that stand in a synonymy, subclass/superclass or part-whole relation to their noun phrase antecedent, and show how the original anaphora resolution algorithm of PENG Light can be extended in a systematic way in order to take care of these bridging definite descriptions. The solution to this problem also sheds some light on the adequate treatment of part-whole relations in a controlled natural language context.

1 Introduction

Computer-processable controlled natural languages such as Attempto Controlled English [12] and PENG Light [34] are **engineered** subsets of natural languages designed to get rid of ambiguity and vagueness that is inherent in full natural language. These controlled natural languages **look like** natural languages such as English but are in fact **formal** languages that can be translated unambiguously into the input language of an automated reasoner and used for several reasoning tasks, among them question answering. Similar to unrestricted natural language, these controlled natural languages allow for anaphoric expressions but their form and usage are considerably restricted. An anaphoric expression (= anaphor) is a word or a phrase that points back to an expression (= antecedent) that has been previously introduced in the text (see [19] for an overview). The two most important types of anaphoric relations that are used in sentences and between sentences in controlled natural languages are pronominal anaphora and definite

noun phrase anaphora. Definite noun phrase anaphora take the form of definite descriptions and proper names. Computer-processable controlled natural languages use relatively “safe” anaphora resolution algorithms that rely mainly on syntactic information, accessibility constraints and the distance of the anaphor to its antecedent [12,29]. This makes it easy for the machine to resolve anaphoric expressions automatically but difficult for the human author to remember the approved forms of anaphoric expressions. In the following discussion, I will focus on definite descriptions, in particular on bridging definite descriptions [21], and discuss how the resolution of these anaphoric expressions that requires inference-supporting knowledge can be managed in a controlled natural language context where not all relevant knowledge is available in advance.

The reminder of this paper is organised as follows: In Section 2, I distinguish different usages of definite descriptions that a controlled natural language processor needs to be able to handle in order to process a text and briefly discuss some related research and contemporary theories. In Section 3, I look at a number of important semantic relations that build the foundation for linking bridging definite descriptions to their noun phrase antecedents via inference. In Section 4, I investigate how well an existing linguistic resource (WordNet) supports the resolution of definite descriptions and discuss alternative approaches to construct the relevant background knowledge for anaphora resolution. In Section 5, I show how a first-order model builder can be used for constructing a model that contains the required background knowledge to resolve definite descriptions during the writing process. In Section 6, I discuss how the language processor of the controlled natural language PENG Light communicates with the model builder and how the author can interact with the language processor in order to augment the text with additional background knowledge that is required to resolve anaphoric definite descriptions. In Section 7, I summarise the advantages and benefits of using an interactive approach to anaphora resolution in a controlled natural language context.

2 Definite Descriptions

Definite descriptions are noun phrases of the form *the F* that begin with the definite determiner *the* followed by the remaining noun phrase *F*. This definition captures one widespread use of the term. However, there are many kinds of expressions that have this surface form in unrestricted natural language (for example generics and plurals) but they have semantic properties that are clearly different from definite descriptions, and there are many expressions that have different surface forms (for example possessives) that could count as being definite descriptions [17]. Definite descriptions have played a central role in logic, semantics, and philosophy of language and their (controversial) discussion goes back to Frege [11], Russell [26], Strawson [30], and Donnellan [9].

2.1 Definite Descriptions and Anaphora Resolution

In context of PENG Light [34], we need a flexible approach to process definite descriptions that pays attention to the various ways individuals can be identified. This approach must leave room for information expressed by definite descriptions to be linked directly to previous information, to be linked indirectly via inference to previous information or to be accommodated as discourse-new information during the interpretation process.

In the simplest case, a definite description that is used anaphorically matches syntactically fully or partially with its noun phrase antecedent and constitutes an example of a direct anaphor. For instance, the head of the noun phrase antecedent in (1) directly matches with the head of the anaphor in (2), and it is relatively easy to establish a direct link between the two expressions:

1. An academic who teaches COMP448 in E6A owns a computer.
2. *The academic* supervises Robert Black and leads the LT Center.

However, the relation between the anaphor and its noun phrase antecedent is often more complex than that of identity. The relation may be a synonymy relation as in (3), a subclass/superclass relation as in (4), or a part-whole relation as in (5):

3. *The scholar* supervises Robert Black ...
4. *The educator* supervises Robert Black ...
5. *The mother board* is faulty and the computer does not start up.

These definite descriptions point back to a noun phrase antecedent that has already been introduced in (1), but they are characterised by a different head noun. The resolution of these definite descriptions requires additional ontological knowledge and some specific kind of inference (also known as implicatures). Note that the definite noun phrases in (3) and (4) refer to the same individual as the noun phrase antecedent in (1). But this is not the case in (5) where the discourse referent of *the mother board* is only “associated” with the one of *a computer* previously introduced in (1). Definite descriptions that have a noun phrase antecedent which uses a different head noun and are related to the antecedent by a relation other than identity are called bridging descriptions.¹

For some semantic relations it is less obvious that they can establish a link between a definite description and a noun phrase antecedent than for others. While a bridging definite description like (4) sounds felicitous in the context of (1), the following one sounds odd:

6. ?*The professor* supervises Robert Black ...

The potential noun phrase antecedent in (1) and the definite description in (6) stand in a superclass/subclass relation while the antecedent in (1) and the

¹ Clark [8] also considers direct anaphora as “bridging”, but the implicature required for resolving these direct references is straightforward.

definite description in (4) stand in a subclass/superclass relation. A possible explanation for the infelicity of (6) is that a subsequent definite description can **in general** not introduce new information about an existing referent (although unrestricted natural language offers of course other ways to do this), so using a more general expression to establish a bridge as in (4) is fine, but using a more specific one as in (6) is unusual and would require additional contextual support. In other words, the definite description in (6) has more descriptive content than the one in (4), and it is this fact that makes accommodation [16] more likely than bridging. That means if no suitable noun phrase antecedent can be found for a definite description, we need to jump into repair mode and accommodate the use of the definite description in a similar way as if this expression had been introduced by an indefinite noun phrase. This process will result in a new discourse referent that we can anaphorically refer to by subsequent noun phrases provided that we take a number of accessibility constraints into consideration (for details see Section 6).

2.2 Related Research and Theories

Hobbs et al. [13] use weighted abduction to interpret discourse and to resolve anaphoric references. Abduction is inference to the best explanation and explains the data at least cost from a knowledge base. Bos et al. [6] develop a theory of bridging for unrestricted natural language in the context of discourse representation theory [14] by extending van der Sandt's theory of presupposition projection [27] with lexical knowledge that is available in form of a rich generative lexicon [24]. Asher and Lascarides [1] show that this theory has shortcomings to model bridging inferences in the absence of presupposition triggers and that lexical semantics is not enough for modeling bridging. They present a sophisticated theory that demonstrates how rhetorical information interacts with compositional and lexical semantics during the update of segmented discourse representation structures and see bridging inferences as a byproduct of this discourse update process. Baumgartner and Kühn [3] develop an abductive solution to resolve anaphoric references in a model generating framework that is in theory able to deal with several discourse histories but the answer to the question how to select between alternative histories is left open.

It is important to note that all these theories assume unrestricted natural language as object of research. The authors gloss over the problem of parsing natural language and take the availability of the relevant (background) knowledge sources for granted.

3 Tailoring Semantic Relations

The resolving of bridging references requires domain-specific terminological background knowledge since the noun phrase antecedent and the anaphor can stand in a number of different semantic relations (see [8] for an overview). I will focus here on three important types of semantic relations: synonymy, subclass

and part-whole relations. Both subclass and part-whole relations are inclusion relations that are transitive and structure the semantic space in a hierarchical fashion. While transitivity of the subclass relation supports valid syllogistic inferences, this does not always seem to be the case for part-whole relations and requires a closer investigation. Before we do this, let us first have a look at how synonymy relations are handled in our controlled natural language context and then progress to the thornier issues that inclusion relations present.

3.1 Synonymy Relations

In unrestricted natural language, synonyms are different words or phrases with identical meaning (= strict synonyms) or very similar meaning (= near-synonyms). In PENG Light (see [34] for an introduction), we only allow for strict synonyms where a word or phrase has exactly the same meaning as another expression. Synonymy relations are binary relations that are transitive, symmetric and reflexive. Noun phrases that use these synonyms have always the same discourse referent. In PENG Light, the author can specify a synonymy relation via a definition that asserts the identity between two expressions, for example:

7. Every scholar is defined as an academic.

It is the author who decides whether two expressions can stand in a synonymy relation or not, and there is nothing that can hinder the author to define, for example, *wumpus* as a synonym of *academic* – but of course this is bad style, and it should be avoided. Another problem that we face is that more than one candidate antecedent might occur in the text – as illustrated in (8) and (9) – that can serve as an antecedent for a definite description:

8. An academic A leads the LT Centre.
9. An academic B teaches COMP249.
10. *The scholar* supervises Robert Black ...

I will discuss a solution to this problem in Section 6 where I show how the ordering of the sentences affects the anaphora resolution process in PENG Light.

3.2 Subclass Relations

A subclass relation (or class inclusion) is often expressed in unrestricted natural language in the form: *An A is a B* or *An A is a kind of B* whereas *A* is referred to as the specific entity type and *B* the generic entity type. In linguistics, *A* is called a hyponym and *B* a hypernym. A hyponym is a word whose semantic range is included within that of another word, its hypernym. In PENG Light, we distinguish between subclass relations such as in (11) and (12), and class membership such as in (13):

11. Every professor is an academic.
12. Every academic is an educator.
13. Anna Grau is a professor.

Note that in contrast to class inclusion in (11) and (12), the class membership relation in (13) relates an individual to an entity type. In the case of class inclusion, we can make the underlying universal quantifier explicit on the surface level of the controlled natural language and use for this purpose the form *Every A is a B* instead of the more implicit version *An A is a B*.

Class inclusion relations are usually understood to express strict partial ordering relations. Strict partial ordering relations are transitive, irreflexive, and antisymmetrical. We can express these semantic properties directly in PENG Light via two conditional statements:

14. If X is a subclass of Y and Y is a subclass of Z
then X is a subclass of Z.
15. If X is a subclass of Y then X is not equal to Y.

3.3 Part-Whole Relations

A part-whole relation (or meronymic relation) has similar semantic properties as the class inclusion relation: it is a transitive, hierarchical, inclusion relation and can provide structure to the terminology. In English, the term *part of* is the most general of a large number of terms that can be used to express various kinds of meronymic relations [35]. The vagueness and generality of this term glosses over more specific meronymic relations and creates problems with transitivity and, as a consequence, makes it difficult to resolve bridging definite descriptions. Here is an example (in unrestricted natural language) that illustrates this problem:

16. A mother board is part of a computer.
17. A ship is part of a fleet.

In (16), the *part of* relation describes a relation between an object (*computer*) and one of its components (*mother board*). This component stands in a specific functional relation to the whole. This is in contrast to (17) where the meronymic relation describes a relation between a member (*ship*) and a collection (*fleet*). Membership in a collection is determined by spatial proximity (or a social relation) between the whole and its parts but does not show a functional relation to the whole. On the other side, membership in a collection and membership in a class (as discussed in Section 3.2) differ in that class membership is based on the similarity of its elements. We can clearly distinguish between (16) and (17) on the surface level of the controlled natural language by replacing the term *part of* in both cases by a more specific term (additionally, we can make the universal quantifier in the subject position of the statement explicit, in a similar way as we have done this for subclass relations before; note that *every* and *each* are predefined function words in PENG Light and have the same meaning):

18. Each mother board is a component of a computer.
19. Each ship is a member of a fleet.

Meronymic relations appear to be transitive but transitivity often seems to fail in “meronymic syllogisms” if the *part of* relation is used in an incompatible way in unrestricted natural language; compare (20) with (21):

20. A CPU is part of a mother board.
A mother board is part of a computer.
A CPU is part of a computer.
21. An engine is part of a ship.
A ship is part of a fleet.
*An engine is part of a fleet.

In (20), we can observe that the *part of* relation describes a semantic relation between a component and an object. But this is not the case in (21) where two different types of meronymic relations are used: the first one describes a relation between a component and an integral object and the second one a relation between a member and a collection. As a result of this mix, the part-whole relation is not transitive and the inference is not valid. We can make the cause of this problem transparent on the surface level of the controlled natural language by replacing the general term *part of* by the two more specific expressions *component of* and *member of* that clearly distinguish between the intended interpretations:

22. Each CPU is a component of a mother board.
Each mother board is a component of a computer.
Each CPU is a component of a computer.
23. Each engine is a component of a ship.
Each ship is a member of a fleet.
*Each engine is a (component OR member) of a fleet.

In [7,35], Chaffin and his colleagues present a taxonomy that distinguishes seven types of part-whole relations: 1. component-integral object (*pedal – bike*), 2. feature-event (*trapeze act – circus*), 3. member-collection (*ship – fleet*), 4. portion-mass (*slice – pie*), 5. phase-activity (*paying – shopping*), 6. place-area (*Everglades – Florida*), and 7. stuff-object (*steel – car*). These part-whole relations can be described via four relation elements which characterise the nature of the connection between the part and whole: (a) whether the relation of a part to the whole is functional or not; (b) whether the part is separable from the whole or not; (c) whether the parts are similar to each other and to the whole or not, and (d) whether the whole possesses (most of) its parts at the same time. Furthermore, the authors show that this taxonomy can explain quite accurately the failure of transitivity in “part-whole syllogisms” and that humans **can distinguish** between these semantic relations if they need to do so.

We have taken the semantic relations (1-4, 6 and 7) of this meronymic taxonomy as a **starting point** for the treatment of part-whole relations in PENG Light. The *phase-activity* relation (5) is currently not used since PENG Light does not allow to anaphorically refer back to a verbal event. For each other type in the taxonomy, we use a default term in controlled language to clarify the semantic relation so that we can better support the resolution of bridging definite descriptions. As a consequence, the term *part of* is not allowed in PENG Light and needs to be replaced always by a more specific term. The following examples (24-29) list these default terms which overlap with a set of expressions that have been frequently used in an empirical relation naming experiment [7]:

24. **Component-Integral Object**

ENG: A CPU is part of a computer.

CNL: Each CPU is **a component of** a computer.

25. **Feature-Event**

ENG: A self-timer is a feature of a digital camera.

CNL: Each self-timer is **a feature of** a digital camera.

26. **Member-Collection**

ENG: A computer is part of a domain.

CNL: Each computer is **a member of** a domain.

27. **Portion-Mass**

ENG: A buffer is part of a memory.

CNL: Each buffer is **a portion of** a memory.

28. **Area-Place**

ENG: A partition is part of a hard disk.

CNL: Each partition is **an area of** a hard disk.

29. **Stuff-Object**

ENG: Aluminum is part of a hard disk.

CNL: Each hard disk is **made of** aluminum.

These default terms can be related to a number of user-defined synonyms in PENG Light. That means each synonym set uniquely identifies a type in the above-mentioned meronymic taxonomy.

4 Knowledge Sources for Anaphora Resolution

WordNet [10] has been used as an approximation of a knowledge base for resolving bridging definite descriptions in unrestricted texts [21,22,23,32]. WordNet groups English words into sets of synonyms and specifies various semantic relations between these synonym sets. These semantic relations depend on the type of the word and include for nouns – among other things – hyponymy/hypernymy and holonymy/meronymy relations. WordNet distinguishes only three subtypes of meronymic relations: component part (*part of*), member-collection (*member of*) and substance (*substance of*). In WordNet, we can, for example, find the information that *computer* is a part of *platform*, that *professor* is a member of *staff* and that *oxygen* is a substance of *air*. That means we can only find information about semantic relations of nouns in WordNet that belong to three of those six meronymic types that are currently available in PENG Light.

WordNet proved to be not a very reliable resource for the automatic identification of correct semantic relations (synonymy, hyponymy, and meronymy) and consequently for the resolution of anaphoric references in unrestricted texts. In almost 40% of the cases, no semantic relation could be found that relates a definite description to its antecedent [21,32]. The situation gets even worse if we

work in an application domain where a very specific vocabulary is required and where the semantic relations between the entries of the vocabulary need to be established first.

Alternatively, we can try to construct a (linguistically motivated) formal ontology for a particular application domain that contains the required terminological knowledge for resolving definite descriptions. Note that this terminological knowledge can be specified directly in PENG Light and then be translated automatically into an expressive description logic [2]. In this scenario, the knowledge base can immediately be used for resolving anaphoric definite descriptions by checking if a discourse referent already exists in the knowledge base and if this discourse referent stands in an approved semantic relation to the information expressed by the definite noun phrase. The information in the knowledge base can even be used to guide the writing process in a predictive way (using similar techniques as in [15,28]) since all background information has been carefully specified in advance.

However, there exists another scenario where a domain expert might want to assert new factual information, but the terminological knowledge is **not yet** available. For example, the domain expert might want to assert first the sentence (1) (see Section 2.1) and then the sentence (4), but the correct resolution of the definite description in (4) would require additional terminological information that probably does not yet exist. This suggests an approach where the domain expert supports the anaphora resolution process and specifies additional terminological knowledge while a text is written. In the next section, I look into the suitability of a state-of-the-art first-order model builder for providing the required background knowledge to support such an interactive approach, in particular, because I am interested in using the expressivity of full first-order logic for knowledge representation rather than a version of description logic.

5 Model Generation

Model generation was introduced as a new reasoning paradigm in the mid-1980's and is able to handle a wider range of specifications than the traditional refutation-based paradigm [18]. Surprisingly, model generation received only little attention within computational semantics and the broader field of language technology although automatic model builders have very attractive properties: they provide a positive way to deal with the satisfiability problem and are able to construct concrete flat models – if there is one – for first-order theories (see [6,25] for an introduction).

5.1 E-KRHyper

PENG Light currently uses E-KRHyper [20] as a reasoning service. E-KRHyper is a general purpose theorem prover and model generator for first-order logic with equality and has been designed for the use in knowledge representation applications (supporting arithmetic evaluation and non-monotonic negation) [4].

E-KRHyper can be seen as an extension of a tableaux-based theorem prover. Tableaux-based provers can be used in two different ways: on the one hand, they can be used for proving that some goal follows from a set of axioms by adding the negation of the goal to the axioms and showing that the resulting set of formulas has no models; on the other hand, they can also be used for model generation, by adding the goal itself to the axioms and showing that the resulting tableau has an open branch which forms a model consisting of a set of ground formulas.

E-KRHyper accepts a first-order logic specification in clausal form (Protein format). That means the output of PENG Light – in our case a discourses representation structure – needs to be translated first into a first-order formula and then with the help of the TPTP tools [31] into the Protein format before the theory can be processed by E-KRHyper. E-KRHyper then computes a E-hyper tableau working always on a single branch. If the computation of a branch reaches a fixed point, then a model has been found. The Herbrand model that E-KRHyper outputs consists of all ground instances of the derived formulas. Since the derived formulas must not necessarily be ground, they characterise in some cases an infinite model (see [4,20,33] for details).

5.2 Model Building for Text Understanding

Models can make correct predictions about the accessibility of bridging definite descriptions provided the information derived from the text is represented in a suitable way. Let's illustrate this claim and start the discussion with a case where the semantic relation between the definite description and the noun phrase antecedent is a synonymy relation. The first sentence in the following mini-discourse (30) asserts terminological information, the second sentence specifies information about a particular application domain, and finally the subsequent sentence starts with a definite description:

30. Every scholar is defined as an academic.
An academic teaches COMP448 in E6A.
The scholar ...

In principle, the model builder can take a first-order representation of the first two sentences as input and generate a model that contains the following variable-free atomic formulas:

31. `academic(sk1). scholar(sk1).`
`teaching(sk2, sk1, comp448). location(sk2, e6a).`

This model can be accessed by an anaphora resolution algorithm and be searched for an antecedent that stands in a synonymy relation to the formula `scholar(X)` that has been derived from the definite description in (30). Note that `sk1` and `sk2` are two Skolem constants: `sk1` stands for an individual and `sk2` stands for a teaching event. The variable substitution `X/sk1` returns information about the existence of a suitable discourse referent and licenses the anaphoric interpretation of the definite description.

However, things get a bit more complex when the noun phrase antecedent and the bridging definite description stand in a subclass/superclass relation:

32. Every academic is an educator.
An academic teaches COMP448 in E6A.
The educator ...

If the model builder processes the first two sentences of (32) in a naive way, we will end up with the following model:

33. `academic(sk1). educator(sk1).
teaching(sk2, sk1, comp448). location(sk2, e6a).`

As the result shows, we lose the information that an academic is a kind of educator since the model only contains extensional information. The model (33) does not contain any information about the semantic relation between these two formulas, and it is not possible anymore to distinguish between a synonymy relation and a subclass relation.

PENG Light uses a flat notation that relies on reification and on a small number of predefined predicates. These predicates distinguish between objects, events, states, names, thematic relations and inclusion relations. This flat notation makes it easy to formulate axioms for these predicates. Taking the rules for subclass inclusion in (14 and 15) into consideration, the model builder is now able to generate a model that contains also the relevant intensional information about subclass relations, for example:

34. `object(sk1, academic). object(sk1, educator).
theta(sk3, agent, sk1). theta(sk3, theme, sk2).
event(sk3, teaching). named(sk2, com448).
theta(sk3, location, sk4). named(sk4, e6a).
subclass(academic, educator).`

Given the formula `object(X, educator)` for the definite description in (32), the anaphora resolution algorithm can now search the model for an antecedent and check additionally whether this antecedent stands in a subclass relationship to the anaphor or not. This information can then be used on the interface level to indicate how the definite description has been interpreted by the anaphora resolution algorithm.

In a similar way, we add information about synonymy relations to the model, for example in the case of (31) the atomic formula `synonym(academic, scholar)`. This makes the relation explicit and – as we will see – the checking for synonymy between an anaphor and its antecedent easier.

So far so good –, but this representation is still not optimal for our purpose; in particular, if the model contains more than one potential antecedent. The following mini-discourse (35) contains, for example, two potential antecedents for the definite description:

35. Every academic is an educator.
An academic A teaches COMP448 in E6A.
An academic B supervises Mary on Monday.
The educator ...

The current representation loses the information about the textual ordering of these antecedents (see the two object-predicates in bold face in (36)). The anaphora resolution algorithm could rely on the numbering convention that is used for naming the Skolem constants but this does not provide adequate information about the distance between the candidate antecedents and the anaphor:

36. `object(sk1, academic). object(sk1, educator).`
`theta(sk3, agent, sk1). theta(sk3, theme, sk2).`
`event(sk3, teaching). named(sk2, com448).`
`location(sk3, sk4). named(sk4, e6a).`
`object(sk5, academic). object(sk5, educator).`
`theta(sk7, agent, sk5). theta(sk7, theme, sk6).`
`event(sk7, supervising). named(sk6, mary).`
`location(sk7, sk8). named(sk8, e7b).`
`subclass(academic, educator).`

We solve this problem by adding an additional argument to each formula that has been derived from a content word. This argument contains information about the offset position of the word in the text. This additional information makes it easy for the anaphora resolution algorithm to select the most recent noun phrase antecedent from a number of options (note: we will not display this auxiliary argument anymore in the subsequent examples):

37. `object(sk1, academic, '8').`
`object(sk1, educator, '8'). ...`

As we have already seen in Section 2.1, superclass/subclass relations between a noun phrase antecedent and a definite description do usually not establish a semantic bridge for anaphora resolution since these definite descriptions introduce new information. Note that the model builder does not generate any information for superclass/subclass relations as long as the preconditions of such terminological statements are not fulfilled. The following example (38) illustrates this case:

38. Every academic is an educator.
 Every professor is an academic.
An academic teaches COMP448 in E6A.
The professor ...

Here only the rule for the subclass relation in the first sentence is triggered but not the one for the subclass relation in the second sentence since the third sentence does not contain an instance of a professor. This results in the following model (39):


```

39. object(sk1, academic).
   theta(sk3, agent, sk1). theta(sk3, theme, sk2).
   event(sk3, teaching). named(sk2, comp448).
   theta(sk5, location, sk6). named(sk6, e6a).
   subclass(academic, educator).

```

That means there is no atomic formula for *professor* available in the model that could serve as an antecedent for the definite description, and therefore a new discourse referent needs to be introduced in this case.

As explained in Section 3.3, the *part of* relation is polysemous and therefore not allowed in PENG Light; as a consequence, the author has to select one of the available alternatives. For all these alternatives, axioms exist similar to those in (14 and 15) that specify these meronymic relations as transitive, irreflexive and antisymmetrical. Let us assume that these axioms are available in PENG Light and used together with the following mini-discourse in (40):

```

40. Each mother board is a component of a computer.
   Anna Grau owns a computer.
   The mother board ...

```

In this case, the model builder generates the model in (41) for the second sentence:

```

41. named(sk1, anna_grau).
   theta(sk3, theme, sk1). theta(sk3, theme, sk2).
   state(sk3, possessing). object(sk2, computer).

```

Note that the rules for the meronymic relation will be only triggered after the definite description in (40) has been processed and after a new discourse referent has been introduced for this definite description. This results in two additional atomic formulas in the model: one that contains the information about the definite description (`object(sk4, mother_board)`) and one that contains the information about the componenthood (`component(mother_board, computer)`).

Now let us have a look at a sentence that contains a disjunction:

```

42. A politician meets ?an academic or talks to ?a student.
   The scholar ...

```

The first sentence has two models and both models provide potential discourse referents; however, disjunction should block anaphoric reference. E-KRHyper generates the first model and stores which formulas have been derived from the selected disjunct (e.g.: `selected_disjunct(object(sk2, academic), _)`). This information can then be used to exclude discourse referents that occur under disjunction as candidate antecedents (see Section 6.3).

Note that in the case of a conditional sentence a model is first constructed for the antecedent part taking the existing context into consideration in order to be able to bind anaphoric expressions in the consequent of the conditional.

6 Anaphora Resolution in PENG Light

PENG Light is a computer-processable controlled natural language that can be used for knowledge representation [34]. The language processor of the PENG Light system translates texts incrementally into TPTP notation [31] with the help of discourse representation structures (DRSs) [14]. The grammar of PENG Light is written in a DCG-style notation that is transformed via term expansion (a well-known logic programming technique) into a format that can be processed by a chart parser. Since the processing of the input is incremental, the chart parser can provide lookahead information that informs the author which word forms or which categories of words can follow the current input string [15]. The language processor of PENG Light communicates with the model builder E-KRHyper for various reasoning tasks. The model builder can be used, among other things, for consistency checking whenever a new sentence has been added to the text or for question answering whenever the author wants to inspect an emerging or existing textual specification [29]. The language processor does not only communicate with the model builder on the discourse level but also on the phrasal level whenever the extended anaphora resolution algorithm (see Section 6.3) needs to access and reason with the terminological knowledge.

6.1 DRSs in PENG Light

In PENG Light, DRSs are built up incrementally during the parsing process using difference lists. In our case, a DRS is a term of the form **drs**(**U**,**Con**) whereas **U** is a list of discourse referents that stand for the entities introduced in the discourse and **Con** is a list of simple or complex conditions for these discourse referents. Simple conditions consist of predicates or equations while complex conditions are built up recursively from other DRSs with the help of logical connectives. The key idea underlying discourse representation theory [14] is that an anaphor can only refer to a discourse referent that occurs in the current DRS or in a DRS immediately superordinate to the current DRS but not to a discourse referent in an embedded DRS. These restrictions make correct predictions about the accessibility of antecedents to anaphora and make DRSs a convenient intermediate knowledge representation format. DRSs can be translated in linear time into TPTP notation (or into any other first-order notation).

The grammar of PENG Light contains not only feature structures for DRSs but also feature structures for syntactic and pragmatic information. The construction of a DRS always runs in parallel with the construction of a syntax tree and a paraphrase. The paraphrase clarifies how the input has been interpreted by the grammar and shows the author all relevant substitutions and provides additional explanations. Some of the conditions in the DRS are annotated with syntactic information in order to improve their processability by other system components. These conditions have the form **Pred#Anno** whereas **Pred** is a predefined predicate and **Anno** is a list that contains syntactic information. In the case of object-predicates, these annotations consist of syntactic information about

person, number and gender of the noun and information about the form of the surface string, for example:

```
43. object(X,mother_board,23)#[[third,sg,neut],[mother,board]]
```

This information can directly be used by the anaphora resolution algorithm that operates over the DRS and communicates with other components of the system (for example the model builder).

6.2 The Original Anaphora Resolution Algorithm

The original anaphora resolution algorithm resolves an anaphorically used noun phrase with the most recent accessible noun phrase antecedent that matches fully or partially with the anaphor and that agrees in person, number and gender with that anaphor [29]. This algorithm has similar coverage as the anaphora resolution of Attempto Controlled English (ACE) [12] and cannot resolve bridging definite descriptions, since it is syntax-based. However, in contrast to ACE where the anaphora resolution algorithm is activated only after the entire DRS for a text has been constructed, the algorithm of PENG Light is embedded into the grammar and triggered whenever a definite expression has been processed.

Representations for definite expressions are built up in an anaphoric DRS that can be seen as a temporary store within the main DRS. This anaphoric DRS consists of unresolved discourse referents (in our case variables) and conditions that contain these discourse referents. Once the processing of a definite expression is complete, the anaphoric DRS is resolved against the main DRS taking syntactic constraints, accessibility constraints and the distance between the anaphor and the antecedent into consideration. If the anaphora resolution algorithm finds a suitable antecedent, then the anaphoric discourse referent Y is related to the accessible discourse referent X via an equation of the form $Y=X$ and this equation is added locally to the main DRS. If no suitable antecedent for an anaphoric expression can be found in the main DRS, then the discourse referent and the corresponding conditions are accommodated as highly as possible to the main DRS. Note that non-anaphoric definite descriptions are processed in PENG Light exactly in the same way as indefinite noun phrases. The result of the anaphora resolution process is always reflected in a paraphrase and immediately visible for the author.

6.3 The Extended Anaphora Resolution Algorithm

The extended anaphora resolution algorithm takes care of bridging definite descriptions and modifies the original algorithm of PENG Light in a systematic way. This new algorithm relies on interactivity and allows the author to specify semantic relations between a noun phrase antecedent and a bridging definite description as part of the specification if this information is not already available. This solution is compatible with the predictive authoring approach that is supported by PENG Light since the anaphora resolution process is machine-guided and the author selects among a number of options [15].

In a first step, the new algorithm searches for syntactic matches in the main DRS in a similar way as the original algorithm, selects the most recent accessible discourse referent but additionally collects all other candidate antecedents in a list. For example, the mini-discourse in (44) contains two possible noun phrase antecedents:

44. An academic A teaches COMP448 in E6A.
An academic B supervises Mary on Monday.
The academic ...

The anaphora resolution algorithm selects the most recent discourse referent as antecedent and replaces the definite description by the noun phrase antecedent in the paraphrase:

45. { The academic B } ...

If the author is not happy with this solution, then he can access one of the other candidates in the list via a keyboard shortcut. Each repeated keystroke replaces the current solution with an alternative binding and updates the main DRS. This basically allows the author to iterate through the list of candidate antecedents and view the result immediately in the paraphrase.

As explained above, the anaphora resolution algorithm of PENG Light is embedded into the grammar and triggers whenever a noun phrase has been processed. In order to process bridging definite descriptions, the algorithm communicates with the model builder and this process can be interpreted as a form of question answering whereas the anaphoric DRS forms the query and the model supplies the facts. If the conditions in the anaphoric DRS do not fully or partially match with the accessible conditions in the main DRS, then the anaphora resolution algorithm sends the main DRS first to the DRS-to-TPTP translator and subsequently to E-KRHyper. The DRS-to-TPTP translator takes care of the skolemization and stores the information about the substitution of variables (e.g. $X/sk1$). This is an important technical point, since the anaphora resolution algorithm needs to remember how these variables have been replaced because the main DRS will be updated according to the outcome of the anaphora resolution process. E-KRHyper takes the skolemized formula as input and tries to construct a model. The output of E-KRHyper is loaded into the Prolog knowledge base and is then available for further processing. The anaphora resolution algorithm can then search the model for a suitable noun phrase antecedent. Since the discourse referents in the anaphoric DRS are represented as variables, the anaphora resolution algorithm has to work with a copy of these variables in the query. As already discussed in Section 5.2, the model can make correct predictions about the accessibility of discourse referents.

Once the model is available, the anaphora resolution algorithm checks first if the anaphor and the noun phrase antecedent stand in a synonymy relation and then if they stand in a subclass relation. Let us assume that the mini-discourse in (46) has just been processed and let us further assume two different situations: one where the author continues the discourse with the definite description

The scholar that stands in a synonymy relation to the noun phrase antecedent *An academic*, and one where the definite description *The educator* stands in a subclass relation to the same antecedent:

46. Every scholar is defined as an academic.
 Every academic is an educator.
An academic who teaches COMP448 in E6A owns a computer.

Since we are interested in all solutions, the anaphora resolution algorithm uses the `setof/3` predicate of Prolog and searches first for synonymy in both cases. The search for synonymy is successful in the first case but not in the second case. Therefore, the anaphora resolution algorithm continues in the second case and searches for a subclass relation between the anaphor and a noun phrase antecedent. The following examples (47) illustrate how the `setof/3` predicate is used to check for subclass relations:

47. `setof([X,P], (subclass(C,educator), object(X,C,P),
 \+ selected_disjunct(object(X,C,P))), Res).`

Note that the search for synonymy is done in a similar way exploiting potential synonymy relations and constraints in the model. In both cases, the relevant information about the offset position and the Skolem constant is collected. If a solution is found, then the main DRS is updated and the paraphrase will reflect the binding via an explanation displayed in square brackets, for example:

48. The scholar [is a synonym of an academic] ...
 49. The educator [is a subclass of an academic] ...

Additionally, all alternative solutions are collected in both cases and the author can – as explained above – jump into “relinking” mode and iterate over the alternatives in the list. Each iteration will update the DRS and generate the corresponding paraphrase.

If no antecedent can be found, then the conditions in the anaphoric DRS are accommodated and a new model is generated. That means a bridging definite description that stands in a meronymic relation to its antecedent is treated in the same way as an indefinite noun phrase and results in a new discourse referent but is marked as “associated” with another term in the paraphrase once a corresponding part-whole relation has been found in the model:

50. Each mother board is a component of a computer.
 Anna Grau owns a computer.
 The mother board [is a component of a computer] ...

If the author is not happy with a solution, then he can specify a suitable semantic relation as part of the discourse on the interface level. Let us assume that the author continues the discourse in (50) with the definite description *The CPU* but that the relevant background knowledge is not yet available:

51. Each mother board is a component of a computer.
Anna Grau owns a computer.
The CPU ...

In this case the author has to switch to the “terminology insertion” mode via a keyboard shortcut that suspends the processing of the current sentence and accepts terminological statements on the fly, for example:

52. Each CPU is a component of a mother board.

Once the terminological information has been specified, the author can switch back to the normal input mode via another keyboard shortcut. The inserted terminological statement is processed as a separate DRS and added to the main DRS. Once this has been done, the model builder updates the model, and the new bridging definite description is licensed in the discourse.

53. Each CPU is a component of a mother board.
Each mother board is a component of a computer.
Anna Grau owns a computer.
The CPU [is a component of a computer] ...

The resulting paraphrase (51) then illustrates the outcome of this knowledge acquisition process, and the author can continue with the writing process.

7 Conclusions

In this paper I showed that anaphora resolution in existing controlled natural languages is quite restricted and argued that we can allow for a wider range of anaphoric expressions if we consider the anaphora resolution process as an interactive knowledge acquisition process. I focused, in particular, on bridging definite descriptions and illustrated what kind of inference-supporting background knowledge is required in order to establish a bridge between an anaphor and a suitable noun phrase antecedent. I discussed three important types of semantic relations (synonymy relations, subclass relations and part-whole relations) that are required in order to support the resolution of bridging definite descriptions. I showed that the naive treatment of part-whole relations results in invalid inferences and therefore potentially wrong resolutions of bridging references. To solve this problem the current implementation of PENG Light uses a more fine-grained representation of meronymic relations than is supported in WordNet. I further argued that a model builder can not only be used as a tool for consistency checking or question answering in a controlled natural language context, but also for resolving anaphoric references that require inference and inference-supporting knowledge. The new anaphora resolution algorithm of PENG Light extends the existing syntax-based algorithm in a systematic way and allows the author to choose among alternative noun phrase antecedents and to add additional terminological knowledge in order to establish an explicit link between an

anaphor and its antecedent while a specification is written. The presented approach fits well together with the predictive interface approach of PENG Light that guides the writing process and allows for adding terminological knowledge in an interactive way. It is important to note that this approach relies on the author of the text – in our case on a domain expert – who works in **collaboration** with the machine and provides the required ontological knowledge while a text is written – this approach brings the human author back into the loop.

Acknowledgments. I would like to thank Norbert E. Fuchs for organising the Workshop on Controlled Natural Language (CNL 2009) on Marettimo Island west of Sicily (Italy), and also three anonymous reviewers of this paper for their constructive and valuable suggestions.

References

1. Asher, N., Lascarides, A.: Bridging. *Journal of Semantics* 15, 83–113 (1998)
2. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.F.: *The Description Logic Handbook*. Cambridge University Press, Cambridge (2003)
3. Baumgartner, P., Kühn, M.: Abducing Coreference by Model Construction. *Journal of Language and Computation* 1(2), 175–190 (2000)
4. Baumgartner, P., Furbach, U., Pelzer, B.: Hyper Tableaux with Equality. In: Pfennig, F. (ed.) *CADE 2007. LNCS (LNAI)*, vol. 4603, pp. 492–507. Springer, Heidelberg (2007)
5. Bos, J., Buitelaar, P., Mineur, A.: Bridging as Coercive Accommodation. In: *Proceedings of CLNLP 1995*, pp. 1–16 (1995)
6. Bos, J.: Exploring Model Building for Natural Language Understanding. In: *Proceedings of ICoS-4*, pp. 41–55 (2003)
7. Chaffin, R., Herrmann, D.J., Winston, M.: An Empirical Taxonomy of Part-Whole Relations: Effects of Part-Whole Relations of Relation Identification. *Language and Cognitive Processes* 3(1), 17–48 (1988)
8. Clark, H.H.: “Bridging”. In: Schank, R., Nash-Weber, B. (eds.) *Theoretical Issues in Natural Language Processing*, pp. 169–174. Cambridge, Massachusetts (1975)
9. Donnellan, K.: Reference and Definite Descriptions. *Philosophical Review* 77, 281–304 (1966)
10. Fellbaum, C.: *WordNet, An Electronic Lexical Database*. MIT Press, Cambridge (1998)
11. Frege, G.: Über Sinn und Bedeutung. In: *Zeitschrift für Philosophie und philosophische Kritik*, NF 100, pp. 25–50 (1892)
12. Fuchs, N.E., Kaljurand, K., Kuhn, T.: Attempto Controlled English for Knowledge Representation. In: Baroglio, C., Bonatti, P.A., Małuszyński, J., Marchiori, M., Polleres, A., Schaffert, S. (eds.) *Reasoning Web. LNCS*, vol. 5224, pp. 104–124. Springer, Heidelberg (2008)
13. Hobbs, J.R., Stickel, M.E., Appelt, E., Martin, P.: Interpretation as Abduction. *Artificial Intelligence* 63, 69–142 (1993)
14. Kamp, H., Reyle, U.: *From Discourse to Logic; An Introduction to Modeltheoretic Semantics of Natural Language, Formal Logic and DRT*. Kluwer, Dordrecht (1993)
15. Kuhn, T., Schwitter, R.: Writing Support for Controlled Natural Languages. In: *Proceedings of ALTA 2008, Tasmania, Australia*, pp. 46–54 (2008)

16. Lewis, D.: Scorekeeping in a language game. *Journal of Philosophical Logic* 8, 339–359 (1979)
17. Ludlow, P.: Descriptions. In: *Stanford Encyclopedia of Philosophy*, July 26 (2007), <http://plato.stanford.edu/entries/descriptions/>
18. Manthey, R., Bry, F.: Satchmo: a theorem prover implemented in prolog. In: Lusk, E., Overbeek, R. (eds.) *CADE 1988*. LNCS, vol. 310, pp. 415–434. Springer, Heidelberg (1988)
19. Mitkov, R.: Anaphora Resolution. In: Mitkov, R. (ed.) *Oxford Handbook of Computational Linguistics*, pp. 266–283. Oxford University Press, Oxford (2003)
20. Pelzer, B., Wernhard, C.: System Description: E-KRHyper. In: Pfenning, F. (ed.) *CADE 2007*. LNCS (LNAI), vol. 4603, pp. 508–513. Springer, Heidelberg (2007)
21. Poesio, M., Vieira, R., Teufel, S.: Resolving Bridging References in Unrestricted Text. In: *Proceedings of ACL 1997 Workshop on Operational Factors in Practical, Robust, Anaphora Resolution for Unrestricted Texts*, July 7–11, pp. 1–6 (1997)
22. Poesio, M., Ishikawa, T., Schulte im Walde, S., Vieira, R.: Acquiring lexical knowledge for anaphora resolution. In: *Proceedings of LREC*, Las Palmas, May 2002, pp. 1220–1224 (2002)
23. Poesio, M.: Associative descriptions and salience: a preliminary investigation. In: *Proceedings of the ACL Workshop on Anaphora*, Budapest, April 2003, pp. 31–38 (2003)
24. Pustejovsky, J.: *The Generative Lexicon: A theory of computational lexical semantics*. MIT Press, Cambridge (1995)
25. Ramsey, A.M., Seville, H.L.: Models and Discourse Models. *Journal of Language and Computation* 1(2), 159–174 (2000)
26. Russell, B.: On denoting. *Mind* 14(56), 479–493 (1905)
27. van der Sandt, R.A.: Presupposition Projection as Anaphora Resolution. *Journal of Semantics* 9(4), 333–377 (1992)
28. Schwitter, R., Ljungberg, A., Hood, D.: ECOLE – A Look-ahead Editor for a Controlled Language. In: *Proceedings of EAMT-CLAW 2003*, May 2003, pp. 141–150 (2003)
29. Schwitter, R., Tilbrook, M.: Meaningful Web Annotations for Humans and Machines using CNL. *Expert Systems* 25(3), 253–267 (2008)
30. Strawson, P.F.: On Referring. *Mind* 59(235), 320–344 (1950)
31. Sutcliffe, G., Suttner, C.B.: The TPTP Problem Library: CNF Release v1.2.1. *Journal of Automated Reasoning* 21(2), 177–203 (1998)
32. Vieira, R., Poesio, M.: An Empirically-Based System for Processing Definite Descriptions. *Computational Linguistics* 26(4), 539–593 (2000)
33. Wernhard, C.: System Description: KRHyper. In: *Fachberichte Informatik 14/2003*, Universität Koblenz-Landau (2003)
34. White, C., Schwitter, R.: An Update on PENG Light. In: *Proceedings of ALTA 2009*, vol. 7, pp. 80–88 (2009)
35. Winston, M.E., Chaffin, R., Herrmann, D.: A Taxonomy of Part-Whole Relations. *Cognitive Science* 11(4), 417–444 (1987)

Talking Rabbit: A User Evaluation of Sentence Production

Paula Engelbrecht, Glen Hart, and Catherine Dolbear

Ordnance Survey, Romsey Road, Southampton, Hampshire,
SO16 4GU, United Kingdom
{Paula.Engelbrecht,Glen.Hart}@ordnancesurvey.co.uk,
Catherine.Dolbear@ordnancesurvey.co.uk

© Crown Copyright 2010 Reproduced by permission of Ordnance Survey

Abstract. Rabbit is a controlled natural language (CNL) designed to aid experts in understanding and authoring domain ontologies. There are three broad types of Rabbit sentence: declarations, axioms and import statements. This paper evaluates the ease with which domain experts without any prior ontology development experience can author declarations and axiom sentences in Rabbit. Participants were asked to author Rabbit sentences about an artificial domain (Planet Zog). The most common error observed was the omission of the every keyword at the beginning of sentences. Another common error was to confuse instance and subclass declarations. Suggested improvements include changes to the Rabbit syntax as well modifications to a CNL authoring environment.

Keywords: Controlled Natural Language, User Evaluation, OWL-DL, Ontology Authoring.

1 Introduction

Ordnance Survey, the national mapping agency of Britain is researching the development of topographic ontologies to describe our spatial data, motivated by a business need to improve ease of understanding, reuse of our data and integration with third-party datasets. Through our experience of authoring topographic ontologies, we have found that the family of Web Ontology Languages (OWL), the W3C standard for ontology authoring, is difficult for domain experts to master. This insight led to the development of a controlled natural language, Rabbit [1], which is designed to make it easier for domain experts – people with expertise in a particular subject area, but not versed in knowledge modeling or Description Logics – to understand and author their own ontologies.

A controlled natural language (CNL) is “a subset of a natural language which is generally designed to be less complex than the complete language and to include only certain vocabulary terms and grammar rules which are relevant for a specific task” [2]. Several CNLs have been developed for and used for ontology authoring. These languages include Sydney OWL Syntax (SOS) [3], Attempto Controlled English (ACE) [4,5] and CloNE [2,6]. With the possible exception of SOS [3], Rabbit differs

from these languages by limiting sentence complexity. Rabbit sentences are relatively simple and describe single concepts. A further difference is that Rabbit, unlike SOS and ACE, does not allow for the use of variables to represent information. Thus, the statement “if X owns Y and Y has Z as a part then X owns Z” is acceptable in both SOS and ACE but not in Rabbit. These design decisions have been made to ensure that Rabbit sentences are easy to comprehend and write. For a comprehensive comparison of Rabbit, SOS and ACE please refer to [7]. The next section will outline the Rabbit syntax in more detail.

2 Rabbit Syntax

The following summary describes the most current version of the Rabbit syntax. It encompasses changes to the grammar that have been informed by a Rabbit comprehension user test reported in a previous paper [8].

Rabbit ontologies consist of sets of sentences. There are three broad types of Rabbit sentence:

1. Declarations: introduce concepts (OWL Classes), instances (OWL Instances) and relationships (OWL properties).
- Concepts are introduced using a concept declaration. For cases where the plural form of a word is irregular (i.e. the plural is not formed by adding an “s” to the end of the singular noun) the plural needs to be added:

<concept name> is a concept [,plural <plural name>].

- For example:

River is a concept.

Sheep is a concept, plural Sheep.

- Subclasses are introduced using “is a kind of”; for example:

Every River is a kind of Waterbody.

- The Rabbit language draws a distinction between core and secondary concepts. Core concepts are central to a given ontology. Core concepts are introduced using a concept declaration; and described using axiom sentences. Secondary concepts are peripheral to a given ontology. For example, the concept “duck” might be required to define the core concept “duck pond” in a hydrology ontology. However, no further information about ducks is needed for the purpose of the hydrology ontology. Secondary concepts are introduced using a concept declaration but not described further.

- Instances are introduced using the key phrase “is a”; for example:

River Thames is a River.

- Relationships are introduced in a similar way to concepts:

<relationship name> is a relationship

- For example:

is the capital of is a relationship.

- It is a Rabbit convention to capitalize concepts and instances but not relationships.
2. Axioms: Axiom sentences describe the declared concepts. There are a number of basic axiom sentences, the most common being:

[Every] <concept> <relationship clause> <object clause>.

- For example:

Every Orchard is a kind of Agricultural Land.
Every Orchard contains a Tree.

- The use of the “every” keyword is optional because sometimes its use would make a statement appear very unnatural, for example:

Every Oil is a kind of Liquid.

- Another basic axiom sentence involves modifying the relationship of a sentence. For example, exactly <n> can be used to specify the number of valid object instances:

Every River has exactly one River Mouth.

- An axiom sentence can also contain an object clause modified by a preposition. Allowable prepositions are currently limited to “that”, “and”, “by”, “for”, and “of”.

Every Church enables Worship by Christians.

- Lists are another common axiom sentence construct. For example:

Every River flows into a Sea or River or Lake or Sink.

- The above sentence denotes that a river can only flow into one of sea, lake, river or sink. This constitutes an implementation of the general Rabbit principle that “or” is treated as exclusive by default. For non-exclusive “or” a relationship modifier (“one or more of”) is needed:

Every River flows into one or more of a Sea or River or Lake or Sink.

- To assert that two things are disjoint the following sentence structure is used:

Rivers and Seas are mutually exclusive.

- Possibly the most complex sentence structure in Rabbit is the one used for stating defined classes (where an equivalence relation is used). Here the sentence comprises a collection of clauses that form a list. For example, the concept ‘City’ could be defined as:

*A City is anything that:
is a kind of Settlement;
has a City Charter.*

The above statement denotes that any settlement that has a city charter is a city.

- Finally, to prevent the construction of complex Rabbit sentences that are difficult to interpret, prepositions cannot be ‘stacked’ in Rabbit. For example, the following somewhat contrived example would not be permitted:

Every Church enables Worship by Christians for Religion.

3. Import statements enable the use of concepts from other ontologies.

The development of Rabbit has been and continues to be an iterative process informed by two sources of information. First, its development has been informed by the building of several medium-scale (approximately 500-concept) ontologies, with updates to Rabbit being based on the requirements that have arisen from these real-world knowledge modeling needs. Second, we have modified Rabbit based on user evaluations: testing domain experts’ understanding and authoring of Rabbit. Rabbit comprehension tests were reported in [8], and this paper now describes the results of our user evaluations of Rabbit authoring.

3 Experimental Procedure and Design

Twenty-one Ordnance Survey employees volunteered to participate in the evaluation. The materials they were given included a short introductory slide show outlining the basic principles of Rabbit and explaining the evaluation task. An information sheet that succinctly outlines 17 essential Rabbit keywords and expressions was also provided. For example, the crib sheet explains that ‘is a kind of’ is used to denote that one class is a subclass of another (e.g. “Every Beck is a kind of Stream”) and that “is a” is used to declare instances. Another example is the use of ‘one or more of’ to specify that the subject can be associated to more than one of the concepts in the list (e.g. “Every River flows into one or more of a River, a Lake or a Sea”). Finally, a short text describing an imaginary domain (the planet Zog and its inhabitants) was provided. This fictitious domain was chosen to control for pre-existing domain knowledge. The participants were given 45 minutes to write as much about planet Zog as they could using Rabbit.

Two sets of analyses were performed. The first analysis consisted of a comparison between the information content participants attempted to capture in their Rabbit sentences and the information content captured by a Rabbit language expert, who

completed the study under test conditions. It is important to note that syntactical accuracy of the Rabbit sentences was not a concern in this analysis. The sentences generated by the domain expert were broken down into 48 units of information, which could be expressed as “subject predicate object” triples of varying complexity (e.g. Bimbles [subject] only exists on [predicate] Zog [object]; Bimbles that breed [subject] always breed [predicate] in a lake [object]; Adult Bimbles [subject] can not [predicate] both work and breed [object]). If a participant managed to capture all 48 units of information then they were given a score of 100%.

The second analysis consisted of an evaluation of the types of errors novice users make when writing Rabbit sentences. The Rabbit sentences were independently scored by three expert users of Rabbit and scores were then compared to ensure scoring mistakes had not been made. In cases where there was disagreement about the presence or type of an error a consensus was reached through discussion. The proportion of sentences that contained errors was calculated for each participant by dividing the number of erroneous sentences by the total number of sentences.

4 Results

The amount of information conveyed ranged widely between participants with a mean score of 60% (StD = 16%), a low score of 29% and a high score of 79%. Performance on individual sentences also varied widely, the highest scoring units of information being mentioned by all participants (100%) and the lowest only being mentioned by 4 participants (19%).

On average, 51% (StD = 22%) of the sentences generated contained at least one Rabbit language error. Again, individual performance varied widely, with the most accurate participant producing errors in 16% of sentences and the lowest scoring participant producing errors in 94% of sentences. Participants’ overall accuracy was quite poor with 50% of sentences generated containing at least one error. Some specific issues were identified:

4.1 Information Content: Core versus Secondary Concepts

In general, the participants were very good at capturing information relating to the main protagonists of the text. For example, all participants mentioned that Bimbles have three eyes and two arms. In fact, all of the sentences that were in the top 25% correct were about Bimbles. It was found that the less related information is to the main protagonists, the more likely it is to be left out. For example, although 67% of participants mentioned that Bimbles don’t eat fish from Lake Quish, only 38% of participants stated that they are not eaten because they are poisonous. Even fewer participants mentioned the chain of causality that causes fish in Lake Quish to be poisonous. An improvement to the experiment might have been to include a scope and purpose for the ontology being built, to ensure that participants knew what in the text was considered important. Nevertheless, these results provide supporting evidence for our use of “Core” and “Secondary” concepts within Rabbit, as the test shows that people naturally categorise information as being more or less important to the central purpose of the ontology.

4.2 Information Content: Omitting Implicit Information

Participants tended to leave out information that is implicit in the name of an entity, demonstrating the difficulty inherent in producing a text that assumes no shared knowledge between writer and reader. For example, only a third of the participants mentioned that Lake Quish is a kind of lake. This omission can be attributed to the fact that people are accustomed to communicate in an environment in which there is some common ground between the listener (reader) and the speaker (writer). Participants were also more likely to mention information which holds for the majority of cases than information which holds for only a few cases. For example, although all of the participants mentioned that “Bimbles are usually red”, only 42% mentioned that blue Bimbles exist. Participants also tended to omit restrictions. For example although the facts that Zoggian fish are killed by fish rot, predation and old age was mentioned by 67%, 47% and 67% of participants respectively, only 33% of participants mentioned that these are the only things that can kill Zoggian fish.

4.3 Syntax Errors: Leaving Out Every

The most common Rabbit error found was the omission of the word ‘every’ at the beginning of sentences. [The use of every was compulsory in the version of Rabbit tested] In fact, if one recalculates the error scores omitting all those sentence in which omission of the word ‘every’ is the only mistake, then the average error rate drops to 43% (StD = 18%). A t test shows that the difference in error percentage between the set of scores that omits ‘every’ only errors and the set of scores that includes them, is statistically significant, $t(19) = 3.11, p < .01$.

4.4 Syntax Errors: Instance versus Subclass

The second most common error was to confuse instance declarations and assertions about one concept being the subclass of another, or to mix elements of both. Participants frequently used ‘is a kind of’ where they should have used ‘is a’ and vice versa. For instance, “Lake Quish is a Lake” would be the correct way to declare that “Lake Quish” is an instance of the “Lake” class. Many participants expressed this type of information incorrectly, by stating, for example, that Lake Quish is a kind of Lake. Similarly, some participants substituted the correct Rabbit sentence “Every Tremit is a kind of Predator” for the incorrect sentence “Every Tremit is a Predator”. The latter sentence contains two errors. First the Rabbit syntax does not allow for the combination of ‘is a’ and ‘every’ in a sentence. Second, ‘is a’ is used where ‘is a kind of’ is required. Removing sentences which only contain this kind of error from the error analysis reduces the overall error rate further from 43% (StD = 18%) to 39% (StD = 18%). This difference was significant, $t(19) = 3.34, p = .01$.

A further type of error which is conceptually related to the above is that participants sometimes used the word “is” to describe a relationship. For example, “A Bimble is usually Red” rather than “A Bimble is usually coloured Red”. Similarly to this type of error, other reserved words in Rabbit, such as ‘no’ and ‘every’ have also been wrongly used, been replaced by a different semantically related word or omitted. For example, the statement “Every Bimble has exactly 1 Tail” has been substituted with “Every Bimble has only 1 tail”. In this sentence the word ‘exactly’ has been

replaced by ‘only’. Similarly, ‘exactly’ has also been omitted entirely. For example, “Every Bimble has a Tail”. This sentence also illustrates another common error, namely, to replace the noun ‘one’ with the indefinite article ‘a’.

4.5 Syntax Errors: Open World Assumption

Participants found it difficult to model knowledge under the open world assumption and understand the meaning of Rabbit keywords ‘only’ and ‘only ... or nothing’. In Rabbit ‘only’ is used to denote that the object of a sentence has to apply to the subject, and that it will be the only object that can be linked to the subject via that relationship. The use of ‘only .. or nothing’ in a sentence, denotes the same as “only” with the concession that the relationship does not have to apply to the subject. The words ‘or nothing’ have been erroneously omitted from the sentence “Every Bimble only breeds in a lake”. This is an error because not all Bimbles breed. Conversely, the sentence “Every Bimble only eats Zoggian Fish or nothing is also incorrect, because every Bimble eats. In fact the text given to the participants specifies that every Bimble has a natural imperative to eat.

5 Discussion

5.1 Information Content

The participants varied widely in the amount of information they conveyed. This variance in performance highlights the need for a software tool to assist with the use of Rabbit. Such a tool (e.g. ROO [9]) could help to ensure that authors who tend to be more restrained in the amount of information they express, include all of the essential information. For example, the tool could ensure that concepts which are introduced as objects in sentences are also defined, if they are considered to be core concepts. Another reason why some participants may have chosen to express relatively little information is that the task of writing Rabbit sentence can be rather tedious. A function that completes items that need to be expressed repetitively (e.g a relationship that is used over and over again) might encourage these individuals to express more information. Furthermore, a software tool would be similarly useful to authors who attempt to convey a lot of information. Here the risk is not so much that central terms remain undefined, but that too many terms are included. The act of defining a scope and purpose for an ontology and the ability to refer to it is supported by the ROO Tool. This functionality will help people to stay focussed on the main theme of what they are expressing. It can act as a constant reminder that some concepts comprise the core of the ontology (and need to be described) whereas others are less central.

Information about the main protagonists of the text (Bimbles and Flugs) was reliably conveyed by almost all of the participants. This finding is unsurprising because it mirrors how people communicate in their day to day interactions. Narratives are usually about a particular topic and the more central information is to this topic the more likely it is to be included. Furthermore, participants tended to omit information that is implicit in the name of an entity and would not need to be explained if one were to tell another person (rather than a machine) about planet Zog. The fact that people are used to communicating with people not with machines

has both benefits and drawbacks for the correct use of Rabbit. Benefits include that people should have an intuitive sense of how to distinguish between primary and secondary concepts. On the other hand, it might cause people to omit important information because it is so implicit that people fail to see that it is not explicit in the ontology they are authoring. The example of “Lake Quish” illustrates this well.

5.2 Rabbit Syntax

The most common Rabbit error found was the omission of the word ‘every’ at the beginning of sentences. This finding combined with the observation that the use of ‘every’ is unnatural in certain contexts (e.g. in combination with some uncountable nouns) has led to the decision to make the use of ‘every’ noncompulsory. This finding highlights the importance of not restricting CNL evaluations to assessments of comprehensibility. Whereas most people would understand what is meant by the ‘every’ keyword, many will forget to include it consistently at the beginning of sentences. Another common error involved the distinction between ‘is a’ to denote instances and ‘is a kind of’ to denote concepts. Participants had difficulties applying this distinction consistently. This finding has led to the decision to drop this distinction and in future both subclasses and instances will be introduced by using ‘is a’. Classes are distinguished from instances because they are declared explicitly. For example, “River is a Concept”. Without this declaration, the sentence “River is a Waterbody” would denote that River is an instance of the class Waterbody. In conjunction with the concept declaration, the sentence “River is a Waterbody” denotes that River is a subclass of Waterbody. The distinction between instances and concepts is further reinforced by the rule that whereas concepts can be preceded by the keyword every, instances cannot. We acknowledge that these changes will not aid authors comprehension of the distinction between instances and classes. However, they should make it easier for those who do understand the distinction to apply the correct construct and to do so consistently.

Another common syntax error was the use of synonyms for reserved words. A possible solution to this problem could be implemented by a rabbit authoring tool. Specifically, the tool could store synonyms for the most common keywords and make the correct suggestion when these are typed in. For example, the software could suggest the use of ‘exactly’ when the word ‘precisely’ is entered.

A final syntax error that needs to be addressed relates to the open world assumption. Some participants omitted the keyword ‘or nothing’ from statements where it was needed. For example, the erroneous statement “Every Bimble only breeds in a Lake” means that every Bimble breeds and that every Bimble breeds in a lake. In order to make it explicit that not every Bimble necessarily breeds the sentence needs to be “every Bimble only breeds in a Lake or nothing”. One possible explanation for this omission is that the inference that every Bimble must breed would not be drawn by a human reader. Overcoming this issue is an area of ongoing research.

6 Conclusion

The aim of the current study was to establish what type of errors novice Rabbit users would make. An analysis of these errors can be used to inform both changes to the

Rabbit language itself as well as to the ROO Protégé plugin (an ontology authoring tool that implements Rabbit) [6]. The usefulness of a software tool is supported by the result that participants varied widely in the amount of information they included in their text. We predict that the use of a software tool would increase the likelihood that important information is included because its use speeds up the process of sentence production. This prediction could be tested in future studies in which Rabbit sentences are authored with the aid of the ROO (or other CNL authoring) tool. Although Rabbit has been designed to be used in conjunction with the ROO tool, participants in the study reported here were not given access to a CNL authoring tool. This was undertaken because we wanted to establish which aspects of the Rabbit language domain experts find difficult to express. The use of a software tool could have masked difficulties users have with using the rabbit language; furthermore, the use of a software tool would have made it unclear whether errors are attributable to an inherent problem with the Rabbit language, or to difficulties with using the tool. The high error-rate observed in this study can be partially attributed to the fact that the participants received minimal instructions and had no software tool to support them. Some of the errors reported here can be avoided by changes to the language, others necessitate tool support.

References

1. Hart, G., Dolbear, C., Goodwin, J.: Lege Feliciter: Using Structured English to Represent a Topographic Hydrology Ontology. In: *Proceedings of the OWL Experiences and Directions Workshop* (2007)
2. Tablan, V., Polajnar, T., Cunningham, H., Bontcheva, K.: User-Friendly Ontology Authoring Using a Controlled Language. In: *Proceedings of the 5th International Conference on Language Resources and Evaluation* (2006)
3. Cregan, A., Schwitter, R., Meyer, T.: Sydney OWL Syntax—Towards a Controlled Natural Language Syntax for OWL 1.1. In: *Proceedings of the OWL Experiences and Directions Workshop* (2007)
4. Kaljurand, K., Fuchs, N.E.: Mapping Attempto Controlled English to OWL DL. In: *Third European Semantic Web Conference. Demo and Poster Session*, pp. 1–12 (2006)
5. Kaljurand, K., Fuchs, N.E.: Bidirectional Mapping between OWL DL and Attempto Controlled English. In: Alferes, J.J., Bailey, J., May, W., Schwertel, U. (eds.) *PPSWR 2006*. LNCS, vol. 4187, pp. 179–189. Springer, Heidelberg (2006)
6. Funk, A., Tablan, V., Bontcheva, K., Cunningham, H., Davis, B., Handschuh, S.: CLOnE: Controlled Language for Ontology Editing. In: Aberer, K., Choi, K.-S., Noy, N., Allemang, D., Lee, K.-I., Nixon, L., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudré-Mauroux, P. (eds.) *ASWC 2007 and ISWC 2007*. LNCS, vol. 4825, pp. 142–155. Springer, Heidelberg (2007)
7. Schwitter, R., Kaljurand, K., Cregan, A., Dolbear, C., Hart, G.: A Comparison of Three Controlled Natural Languages for OWL 1.1. In: *Proceedings of the OWL Experiences and Directions Workshop* (2008)
8. Hart, G., Johnson, M., Dolbear, C.: Rabbit: Developing a controlled natural language for authoring ontologies. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) *ESWC 2008*. LNCS, vol. 5021, pp. 348–360. Springer, Heidelberg (2008)
9. Denaux, R., Dimitrova, V., Cohn, A.G., Dolbear, C., Hart, G.: ROO: Involving Domain Experts in Authoring OWL Ontologies. In: Sheth, A.P., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T., Thirunarayan, K. (eds.) *ISWC 2008*. LNCS, vol. 5318, pp. 1–16. Springer, Heidelberg (2008)

Naturalness vs. Predictability: A Key Debate in Controlled Languages

Peter Clark, William R. Murray, Phil Harrison, and John Thompson

Boeing Research and Technology, P.O. Box 3707, Seattle, WA 98124, USA
{peter.e.clark, william.r.murray}@boeing.com,
{philip.harrison, john.a.thompson}@boeing.com

Abstract. In this paper we describe two quite different philosophies used in developing controlled languages (CLs): A "naturalist" approach, in which CL interpretation is treated as a simpler form of full natural language processing; and a "formalist" approach, in which the CL interpretation is "deterministic" (context insensitive) and the CL is viewed more as an English-like formal specification language. Despite the philosophical and practical differences, we suggest that a synthesis can be made in which a deterministic core is embedded in a naturalist CL, and illustrate this with our own controlled language CPL.

In the second part of this paper we present a fictitious debate between an ardent "naturalist" and an ardent "formalist", each arguing their respective positions, to illustrate the benefits and tradeoffs of these different philosophies in an accessible way.

Part I: The Naturalist vs. Formalist Debate

1 Introduction

There are two quite divergent schools of thought concerning the design of controlled languages. The first, which we call the "**naturalist**" approach, treats CL interpretation as a simpler form of the full natural language (NL) processing task in which ambiguity still resides, only to a lesser extent. One might say that the goal is to make English more tractable (understandable by computers) by simplifying the complexity of the English handled. In this approach, as with full NL, multiple interpretations of a sentence are possible, and the CL interpreter uses all the standard NL machinery to search for a "best" parse and interpretation, but with the task being constrained to just the subset of English covered by the CL. Examples of "naturalist" CLs include our own, called CPL [1] (described later), and (to a degree) CELT [2].

The second approach, which we call the "**formalist**" approach, views a CL more as an English-like formal/programming language that is well-defined, predictable, and easier to use than a normal formal language. One might say that the goal is to make logic more tractable (easier to use by humans) by rendering it in human-readable terms that non-mathematicians can understand. Given that many existing formal languages are somewhat cryptic to untrained users, and good NL processing techniques already exist, it is a natural step to have users work with CLs instead that

are more readable and translate deterministically into the formal language. A "formalist" approach would view this as an end in itself, and make no claims that the CL necessarily says anything about NL processing in general. Examples of "formalist" CLs include ACE [3], PENG [4], CLCE [5], and CLIE [6].

Although it might seem that the "naturalist" and "formalist" approaches are just variants for tackling the same overall problem, their underlying philosophies are quite different. A formalist approach eschews nondeterminism, requiring that the CL translates cleanly and predictably to a formal representation, i.e., there is only one acceptable parse and interpretation for any sentence, there is a single sense for each word (plus part of speech), and interpretation follows a logically defined path. In contrast, a naturalist approach attempts to do NL interpretation "in the small," making many disambiguation decisions heuristically (e.g., prepositional phrase attachment, semantic role labeling), and searching for an overall "best" interpretation. A naturalist might seek to gradually extend the CL with time, taking small steps towards fuller NL understanding, while a formalist might declare the CL complete once sufficient expressivity had been achieved.

These different approaches can produce quite different results. A naturalist CL may be more fluent/natural for the user, but also harder to control because the user cannot always predict the disambiguation decisions that the system will make. Conversely, a formalist CL may be easier to control (once the user has learned the disambiguation rules), but may also be somewhat less natural to read and may require the user to understand more about the target representation language and ontology.

At Boeing we have been developing a controlled language called CPL (Computer-Processable Language) which clearly falls in the "naturalist" category. We have also created CPL-Lite, a more constrained version which (unlike CPL) is deterministic and falls into the "formalist" category. In this paper, we describe these two languages and discuss the strengths and weaknesses of each. We then suggest a way in which the dichotomy between these two rather different approaches might be resolved, and describe ongoing work at creating such a combination.

Following this, the second half of the paper presents a debate between a "naturalist" and a "formalist", each arguing his position. Although the dialog is fictitious, it draws on material from numerous discussions over the years, including at the CNL 2009 Workshop. The goal of presenting the debate is not to argue one position over the other, but rather to highlight the strengths and weaknesses of both philosophies in an easily accessible way.

2 Two Controlled Language Variants

2.1 CPL – Computer-Processable Language

CPL is a mature and extensive controlled language, used in several projects [1,7]. It is clearly in the "naturalist" category as it attempts to do simplified natural language processing, using a variety of heuristics to make disambiguation decisions and produce the "natural" or "obvious" interpretation that a person would expect.

Briefly, CPL accepts three types of sentences: ground facts, questions, and rules. For ground facts, a basic CPL sentence takes one of three forms:

“There is/are NP”

“NP verb [NP] [PP]*”

“NP is/are passive-verb [by NP] [PP]*”

where *verb* can include auxiliaries and particles, and nouns in *NPs* can be modified by other nouns, prepositional phrases, and adjectives. For questions, CPL accepts five forms, the two main ones being:

“What is NP?”

“Is it true that Sentence?”

For rules, CPL accepts the sentence pattern:

“IF Sentence [AND Sentence]* THEN Sentence [AND Sentence]*”

CPL's grammar supports simple sentences, prepositional phrases, compound nouns, ordinal modifiers, proper nouns, adjectives, pronouns, definite reference, and a limited use of conjunction (allowing fillers of the same semantic role to be conjoined, e.g., "John met Sue and Mary"). It does not allow adverbs, modals, relative clauses, imperatives, disjunction, or other forms of coordination. For (non-unitless) physical quantities, e.g., "10 meters", a unit of measure must be given.

Definite reference is resolved by searching for the most recent, previously mentioned object with the same description, e.g., "The red block..." will resolve to the most recently mentioned red block. Ordinal reference, e.g., "The second red block...", behaves similarly except counts referents starting from the beginning of the CPL text. If a referent is not found, CPL assumes the reference is to an implicitly existing object and so creates that object in its internal interpretation. Pronouns resolve to the most recent object with the same gender as the pronoun.

Existential quantification is expressed using CPL sentences, and these always translate to Skolemized (i.e., existential) ground assertions. Universal quantifiers such as "every", "all", "some", and "most" are not allowed. Universal quantification is expressed using CPL "IF...THEN..." rules, and quantifiers are applied in the same way as in Prolog (namely variables in the "action" part of the rule are universally quantified, and remaining variables are existentially quantified).

CPL is designed to work with a pre-defined ontology (including words associated with each concept and relation in the ontology), the primary one being UT Austin's Component Library ontology [8], although others can be used too. Words outside the target ontology's associated lexicon can be used, in which case CPL uses WordNet and the target ontology's lexicon to find the closest concept to those words. CPL does not currently support extending the lexicon or ontology using CPL statements.

The most extensive application of CPL to date has been for posing exam-style questions to AURA [7], a knowledge-based system containing formal knowledge about physics, chemistry, and biology [9,10]. To pose questions, users reformulate the original English in CPL, aided by a set of guidelines, examples, and on-line feedback when invalid CPL is entered. For example, given the original AP exam question:

An alien measures the height of a cliff by dropping a boulder from rest and measuring the time it takes to hit the ground below. The boulder fell for 23 seconds on a planet with an acceleration of gravity of 7.9 m/s^2 . Assuming constant acceleration and ignoring air resistance, how high was the cliff?

A typical way that the user might express this in CPL would be:

- An alien drops a boulder.
- The initial speed of the boulder is 0 m/s.
- The boulder drops for 23 seconds.
- The acceleration of the boulder is 7.9 m/s^2 .
- The distance of the drop equals the height of the cliff.
- What is the height of the cliff?

As can be seen, the translation from the original English to CPL requires some work by the user, both conceptually (to factor out some of complex notions in the original English that are outside the expressive capability of AURA, and to make commonsense knowledge explicit), and linguistically (to simplify the sentence structure). However, given a good set of worked examples, on-line feedback by CPL, and a small (4 hours) amount of training, users were able to master this task during our evaluations [7], although typically making several attempts before finding a valid and adequate CPL formulation.

CPL is implemented as the vocabulary and grammar restrictions described above placed on the underlying language interpreter BLUE (Boeing Language Understanding Engine). Sentence interpretation is performed using a 10-step pipeline:

1. **Preprocessing** – remove non-ASCII characters etc.
2. **Parsing** – using SAPIR, a broad-coverage chart parser [11]
3. **Syntactic logic generation** – convert parse tree to “syntactic” logic where predicates and arguments are words or Skolems, not concepts.
4. **Reference resolution**
5. **Transform Verbs to Relations** – convert some reified verbs to predicates
6. **Word Sense Disambiguation (WSD)** – using WordNet to help pick senses, thus replacing word arguments with concepts from the target ontology.
7. **Semantic Role Labelling (SRL)** – using a hand-crafted set of ~100 rules to replace word predicates with semantic relations.
8. **Metonymy resolution** – using 5 metonymy rules e.g., PLACE for OBJECT
9. **Question Annotation** – tag the query variables
10. **Additional Processing** – miscellaneous tidying up

Further details are given in [12]. Heuristic rules are used for prepositional phrase attachment, word sense disambiguation, semantic role labeling, compound noun interpretation, metonymy resolution, and other language processing activities. Output is in a Prolog-like syntax. A simple example of the input and output is as follows:

```

;;; CPL: "A person throws a ball from the top of a building."
isa(person01,Person),
isa(ball01,Hollow-Ball),
isa(top01,Region),
isa(building01,Building),
isa(throw01,Throw),
has-region(building01,top01),
agent(throw01,person01),
object(throw01,ball01),
origin(throw01,top01).

```

2.2 CPL-Lite

CPL-Lite was designed for a separate project called Mobius [13] to allow trained knowledge engineers, who understood the target knowledge base, to pose queries in a way that was controllable and also (reasonably) comprehensible to others. While CPL searches for the best interpretation, CPL-Lite is simpler and interpreted deterministically (no search or use of heuristics).

CPL-Lite uses the same interpreter as CPL, but with additional restrictions and modifications. First, the grammar is substantially further restricted to just a set of template sentences (described shortly), thus avoiding ambiguities that would otherwise arise. Second, the vocabulary is restricted to just words that map directly to a concept in the ontology (WordNet is not used to "guess" the meaning of other words). Third, CPL's heuristics are switched off and just a default behavior is used: word sense disambiguation (WSD) simply picks the a priori most likely word sense; semantic role labeling (SRL) rules are fixed and defined by the template sentences (below); and metonymy resolution rules are not used. Although these restrictions may seem severe, bear in mind that CPL-Lite is designed to be used by a knowledge engineer with full knowledge of the ontology its vocabulary, rather than an end-user.

For the sentence templates, each CPL-Lite sentence corresponds to a single binary relation (i.e., slot, predicate) between two entities. For assertions, there are 113 sentence templates of three types (below), depending whether the relation appears in language as a noun, verb, or preposition:

<u>CPL-Lite Sentence Pattern</u>	<u>Interpretation</u>
<i>For the 82 noun-like relations:</i>	
"The age of a <i>entity</i> is a <i>duration</i> ."	→ age(<i>entity</i> ', <i>duration</i> ').
"The agent of a <i>event</i> is a <i>entity</i> ."	→ agent(<i>event</i> ', <i>entity</i> ').
...etc...	
<i>For the 10 verb-like relations:</i>	
"A <i>event</i> ₁ causes a <i>event</i> ₂ ."	→ causes(<i>event</i> ₁ ', <i>event</i> ₂ ').
"A <i>entity</i> ₁ encloses a <i>entity</i> ₂ ."	→ encloses(<i>entity</i> ₁ ', <i>entity</i> ₂ ').
...etc...	
<i>For the 21 preposition-like relations:</i>	
"A <i>entity</i> ₁ is above a <i>entity</i> ₂ ."	→ is-above(<i>entity</i> ₁ ', <i>entity</i> ₂ ').
"A <i>entity</i> ₁ is behind a <i>entity</i> ₂ ."	→ is-behind(<i>entity</i> ₁ ', <i>entity</i> ₂ ').
...etc..	

where *entity*' is the interpretation of (i.e., instance denoted by) "a *entity*", etc.

The significance of this is that it allows any of the 113 predicates in the underlying ontology to be expressed unambiguously, and thus allows the user to “force” a particular interpretation in a formalist-style manner (although the CPL-Lite English will thus be more verbose than the corresponding CPL, as only one relation is expressed per sentence). *NP* is one of ~1000 simple nouns (including a few compound nouns) that map directly to concepts in the target ontology, i.e., is in the ontology's associated lexicon. Complex noun phrases are not allowed. In addition, the sentence form “*NP verb [NP]*” is allowed, where *verb* is in the ontology's lexicon (mapping to an action/event concept), and with the interpretation that the first and second NP are always the agent and object of the verbal concept respectively, i.e., are interpreted as agent(*x,y*) and object(*x,y*). Definite reference, “IF...THEN...” rule sentences (containing CPL-Lite sentences only), and three question forms are also supported in CPL-Lite.

3 Comparison, and the Naturalist vs. Formalist Tradeoff

Interestingly, CPL-Lite has the same expressivity as CPL; it is just more verbose and grammatically restricted, and requires more knowledge of the vocabulary and structure of the target ontology. It is also more predictable: A knowledgeable user can enter a sentence and know exactly how it will be interpreted. In contrast, the “naturalist” language CPL is more fluent and tolerant of the user: it uses WordNet to “guess” meanings for unknown words, will use lexical and semantic knowledge to try and perform PP attachment and semantic role labeling correctly, and attempt to resolve metonymy. However, as the CPL interpreter is more complex than that for the “formalist” language CPL-Lite, there is arguably more of a risk that CPL may not interpret the sentence in the way the user intended, and it may not be obvious to him/her how to reformulate the CPL to correct the error (if indeed the user is aware of the misinterpretation). To handle this, CPL paraphrases and graphs its interpretation back to the user so he/she can validate/correct the system's understanding.

To illustrate the naturalist/formalist distinction, consider an example of CPL and the corresponding CPL-Lite, taken from the AURA application [9]. In this example, the user describes part of a physics problem in CPL as follows:

A man drives a car along a road for 1 hour.
The speed of the car is 30 km/h.

When processing this, CPL interprets “for” here to mean the predicate duration(), “along” to mean path(), and attaches both prepositional phrases in the first sentence to the verb (“drive”). In addition, the target ontology considers speeds as properties of events, not objects, and so CPL interprets the second sentence as metonymy for “The speed of the car's *driving* is 30 km/h” (i.e., it resolves the metonymy with respect to the target ontology). Finally, the concepts of “car” and “man” are not part of the CLib ontology, so CPL climbs the WordNet taxonomy to find generalizations which are in the ontology – here Vehicle and Person – and maps these words to those concepts, i.e., tries to find the nearest concept which is known. The resulting interpretation looks:

```

isa(drive01, Drive),
isa(man01, Person),
isa(car01, Vehicle),
isa(road01, Road),
agent(drive01, man01),
object(drive01, car01),
path(drive01, road01),
duration(drive01, [1,hour]),
speed(drive01, [30,km-per-hour]).

```

Note that the same resulting interpretation¹ could have been produced by a CPL-Lite formulation as follows:

```

A person drives a vehicle.
The path of the driving is a road.
The duration of the driving is 1 hour.
The speed of the driving is 30 km/h.

```

In this CPL-Lite formulation, the user had to explicitly identify and spell out the target concepts (e.g., Person); had to explicitly spelled out (the words corresponding to) the target predicates; has removed the PP attachment ambiguity; and has correctly (with respect to the constraints in the target ontology) attached the speed to the driving event (rather than the car) as required by the ontology.

The important point to note is that both formulations produce the same resulting logic. To write in the formalist CPL-Lite style, the user needs a clearer picture of the target representation he wishes to build, i.e., needs to know the specific concepts and predicates he wishes to use, how they can be combined, and what words can be used to refer to them. He also needs to write in a more verbose and, here, somewhat less natural way. However, the authoring task is also quite straightforward – the user writes in simple, unambiguous English and can easily predict how it will be interpreted. In contrast, the “naturalist” CPL has the advantage of perhaps being more fluent, but also carries an increased risk of being misinterpreted. For example, consider the (hypothetical) case that the CPL interpreter misinterprets “for” in the earlier CPL sentence:

A man drives a car along a road for 1 hour.

as meaning beneficiary(x,y) (i.e., the hour is the “beneficiary” of the driving). In this case, CPL’s “smarts” have gone wrong, and the user is left either unaware of the error, or aware of it but unsure how to rephrase the sentence to correct the problem. To mitigate this problem, CPL paraphrases back its understanding in CPL-Lite and also as a graph so the user is aware of that understanding, and provides reformulation advice and a library of CPL examples to help the user reword if necessary. In general, though, “smart” software is a mixed blessing -- it can be very helpful, or frustrating to control, or both (e.g., consider automatic page layout or figure placement software). This is essentially the tradeoff that the naturalist vs. formalist approaches presents. We illustrate these tradeoffs further in the dialog in part two of this paper.

¹ Bar different arbitrary Skolem names. Note man01 is an instance of Person, not man, as there is no concept of a man in the target ontology.

4 Synthesis: Combining the Two Approaches

While it may seem that the underlying philosophies of the naturalist vs. formalist approaches are incompatible, there is a synthesis which we have made, namely to "embed" CPL-Lite as a deterministic core within CPL itself. In other words, within CPL, we have included the core set of the 113 CPL-Lite sentence patterns described earlier whose interpretation is deterministic (i.e., context-insensitive) and easily predictable. As CPL-Lite already uses the same interpreter as CPL, the mechanism for this "embedding" is straightforward, namely to ensure that when the CPL-Lite sentence templates are used, only CPL's default behavior rather than additional heuristics are applied. Thus only a small change to the existing CPL software was needed to ensure that no "fancy heuristics" were triggered by use of the CPL-Lite templates.

Given such a core, the user can then work within it or beyond it to the extent that he/she feels comfortable, and can fall back on the core if the "smart" interpretation goes wrong. For example, should the CPL interpreter have misinterpreted "for" in "drives...for 1 hour" as beneficiary(x,y), the user can revert to CPL-Lite to make the desired relation explicit: "The duration of the driving is 1 hour", thus correcting the mistake. In this way, the user both has the flexibility to write in more fluent English, while also being able to resort to a more constrained form when necessary to control the interpretation.

Over the last two years we have used CPL, including the deterministic CPL-Lite core, in the AURA application described earlier, including evaluating its performance [14]. One striking feature of CPL's usage in this context is that although CPL provides for a wider variety of forms than CPL-Lite, users typically stay within the CPL-Lite subset in the majority (~70%) of their sentences. The most common examples of going beyond CPL-Lite were using complex noun phrases (e.g., "the direction of the applied force on the object"), using words outside the KB's lexicon (e.g., "car", "horse"), and using metonymy with respect to the KB (e.g., "The speed of the man" for "The speed of the man's movement"). As the users were trained with largely CPL-Lite-style example sentences it is perhaps not surprising that they often stayed within this subset, and did not often venture into more sophisticated language forms, and when they did venture out it was somewhat conservatively. This suggests that for users to feel comfortable going beyond that core, the quality of the interpretation needs to be high, and thus "naturalist" CLs are perhaps mainly appropriate for domain-specific applications where the required level of domain-specific customization can be made.

5 Summary

Although the philosophies underlying the naturalist and formalist approaches differ, there is a strong case to be made that each needs and can benefit from the methods of the other. For a naturalist CL such as CPL, there still needs to be a way for the user to control the interpreter's behavior when he/she knows what target output is needed, and this can be done by embedding a formalist-style core, as we have done by embedding CPL-Lite as a core of CPL. Conversely, for a formalist CL such as CPL-Lite, the CL can sometimes be verbose and disfluent, and require a deeper understanding of the

target ontology. Adding a more naturalist-like layer to the CL can alleviate these problems, providing that there is sufficient feedback to the user so that there is no confusion about what the system understood. In fact, in practice there are degrees of non-determinism that might be introduced or rejected (e.g., grammar; word senses; semantic roles), and so in implementational terms there is perhaps more of a continuum between the naturalist and formalist extremes. Thus although there are two quite different philosophies underlying the design of modern CLs, each has substantial benefits to be gained from the other, and there are good reasons for expanding the dialog between practitioners of each.

Part II: A Debate between a Formalist and a Naturalist

As we have discussed, the tradeoffs between the formalist and naturalist positions are quite complex, and there are strong arguments both for and against each position. To make these more vivid, and also capture some of the discussions both before and during the CNL 2009 Workshop, we conclude with a fictitious debate between a "formalist" (**Form**) and "naturalist" (**Nat**) about their respective positions. It integrates several real discussions and exchanges over the years, including from the CNL'2009 Workshop, on the topic of the naturalness vs. predictability tradeoff. The discussion is not meant to reach a conclusion, but rather highlight the points made by proponents of both philosophies in an accessible and balanced way.

[Predictability vs. Naturalness]

Form: Well, as a firm believer in the "formalist" approach I'm not sure I buy all of this. It seems essential to me that CNLs should be *unambiguous*, i.e., for any sentence, there is *just one and only one valid interpretation*. CNLs are, by definition, intended to be a precise means of communication between the user and the machine, and if we allow ambiguity in the process then we will have reintroduced precisely the problem that CNLs are meant to avoid. Parsing, for example, should be deterministic - there should be only one valid parse for any given sentence. In that way, the user has complete control over how his input is interpreted, and can create the target formal representation he intends.

If we allow non-determinism, ambiguity, and heuristics in the CNL interpreter, how is the user ever going to ensure that what he says is interpreted in the way he intends? If we try and make the software too clever, it will only end up confusing the user. Predictability is better than smarts!

Nat: I think the claim of "no ambiguity" is misleading: Ambiguity is a property of language, not of a language interpreter, and it is just a fact that most English statements are inherently ambiguous. Any CNL, formalist or naturalist, has to resolve that ambiguity in some way. If a "formalist" CNL only "sees" a single interpretation, then it still has made a choice, implicitly in its design, not to consider any alternative interpretations. But to the average user those alternatives are still there.

Form: Well, maybe the phrase "no ambiguity" isn't the best, but let me try to be clear what I mean: I believe a CNL should have a "deterministic" grammar, i.e.,

a small set of grammar rules that do not conflict with each other or offer multiple alternatives. The grammar should only ever admit *one* interpretation, so it is completely clear to the user how a sentence will be interpreted.

For example, we might have a rule that prepositional phrases always attach to the preceding verb (an application of the minimal attachment principle). Thus, given a potentially ambiguous sentence, it is now completely clear how the CNL will understand it.

Nat: Completely clear to who? Suppose the user writes:

(1) The man lives in the house with a red roof.

It is "completely clear" to the user that the house has a red roof (rather than that the man lives with a red roof). But a CNL using your attachment rule will not interpret the sentence this way.

Form: Well, the user has to learn the grammar rules, and consider how they will be applied as he authors CNL sentences.

Nat: That's a lot of work for the user!

Form: So what? Users are smart and can be trained!

Nat: But why make things so hard for the user? In the end, all the user wants is to be understood correctly. If he writes:

(1) The man lives in the house with a red roof.

he wants the computer to understand the house has a red roof, while if he writes

(2) The man lives in the house with his wife.

then he wants the computer to understand that the man is living with his wife.

Form: Well, if he wants to say (1) then he should simply phrase the knowledge in a different way, e.g.,

(3) The man lives in the house that has a red roof.

That's not a big deal.

Nat: But how does the user know that (1) needs to be rephrased?

Form: He learns the CNL rules, or we can have the CNL paraphrase its interpretation back to the user² so he can see whether it was understood in the way he intended.

Nat: This seems to be making life very hard for the user. Look, we want to enable the user to write as naturally as possible. While this is too difficult to implement for full natural language, the beauty of CNLs is we can restrict the language to a point where natural authorship and interpretation is possible. Why make the user do this kind of rephrasing (3) you suggest if we can build CNL interpreters that can understand the original (1) in the first place?

² e.g., see Kaljurand's paper [15] elsewhere in these proceedings.

Form: No, I disagree. We want the users to write as precisely as possible, and encourage them to be clear and unambiguous. If the system is full of heuristics, how is the user going to know how any CNL statement he writes will be understood? Suppose the heuristics go wrong? Suppose a statement is misinterpreted?

Nat: Well, as you yourself just suggested, the CNL interpreter can paraphrase back its understanding to the user so he can validate that the computer has understood correctly. We could also use tools to help the user construct only sentences that are valid³. And this requirement for user validation isn't just for a "naturalist" CNL - what happens if a "formalist" CNL doesn't interpret the user's sentence the way he expects, e.g., because he hasn't fully internalized the rules, or the CNL's lexicon is missing something? *Any* usable CNL is going to need some way to allow the user to see and validate/correct the system's interpretation.

Form: I think the user is going to make fewer mistakes using a "formalist" CNL, because he can predict how his sentences will be interpreted.

Nat: No, I would say the opposite is true - I think the user is going to make fewer mistakes using a "naturalist" CNL, because the system's interpretation rules are closer to those of natural language, and so the system is more likely to do what the user naturally expects.

Form: Hmmm....these conjectures would be interesting to test.

Nat: Indeed! Perhaps your approach might best suit users trained in logic where their ultimate task is to write specifications. And mine might best suit lay users who just want to *use* a system by interacting with it in English.

[Lexical Ambiguity and Word Sense Disambiguation]

Form: Here is another important aspect of many "formalist" CNLs: each word (in a given part of speech) should have only one sense - the "one sense per word" principle. This I believe is important, because then the user does not have to worry about word sense disambiguation. Rather, he just needs to use words consistently. The words he uses becomes the ontology of the theory he builds.

Nat: That's a big restriction! That might work fine for nouns and verbs, if the user is careful, but what about for prepositions? For example, given

(4) The roof of the house...

what predicate would "of" translate to?

Form: It just becomes the predicate: of(x,y).

Nat: But what does of(x,y) *mean*?

Form: It means whatever the user says it means. The user would write whatever rules (in CNL) he felt was appropriate to capture the meaning. Again, we want the user to have complete control, rather than be shoe-horned into a pre-defined ontology.

³ e.g., menu-driven sentence builders such as NL-Menu [16], the Ace Editor and AceWiki [17], and ECOLE used in PENG [18].

Nat: But "of" can mean many things, e.g.,:

- (4) The roof of the house... [part of]
- (5) The cellphone of the girl... [possessed by]

"of" clearly corresponds to a different semantic relation in (4) and (5); you can't just translate all of these to: $of(x,y)$! Suppose I want to write axioms about the "part of" relation, for example? Those axioms should apply to (4) but not (5), but if both are translated to $of(x,y)$ then the important distinction is lost.

Form: Well, my first reaction is to say the user shouldn't write such ambiguous sentences in the first place. Instead, he should make the relation explicit, for example instead of (4) and (5) he should write:

- (6) The roof that is a part of the house...
- (7) The cellphone that is possessed by the girl...

Alternatively, the user can write axioms for $of(x,y)$ directly by including extra conditions in those axioms. For example, if the user wants to write axioms for cases where $of(x,y)$ means $is-part-of(x,y)$, he can add conditions to select just those cases where "of" means "part of". He might decide that if x and y are artifacts, then " x of y " means " x part-of y ", and so can write part-of axioms using "of" plus an "artifact" test:

- (8) **IF** there is an artifact X **of** an artifact Y
THEN *<some assertion about part-whole relationships>.*

Nat: Ugh! Your first alternative, (6) and (7), requires the user to mentally do preposition disambiguation himself. How will he even notice when it needs to be done? The second alternative, (8), looks even worse - the user is essentially having to write semantic role disambiguation rules within all his axioms!

Form: Sure, why not? The user *should* be doing the disambiguation himself and *should* be precise in his CNL sentences. How else is he going to reliably communicate with the machine?

Nat: That's placing a huge burden on the user, making him write in a less natural and more convoluted way.

Form: But that's a good thing, if that natural way is unclear or ambiguous.

Nat: No, that's a bad thing - the user would be happier if he could write naturally and have the computer work out the meaning itself! For example, he'd like to write:

- (4) The roof of the house...

and have the CNL interpret this as, say, $has-part(house01, roof01)$.

Form: I think that takes too much control away from the user, and unnecessarily forces him into a particular target ontology.

Nat: Or maybe that's helping the user, as the CNL interpreter is doing some of the work for him.

Form: Or maybe that's not helping the user, if the CNL interpreter's heuristics go wrong. The user will be happiest if the system is easy to control, and if he can easily predict what it will do with his sentences.

[Canonicalization]

Nat: A similar issue occurs with common, highly ambiguous nouns and verbs. Consider the meaning of the verb "be" in the following sentences:

(9) The box is red.

(10) The color of the box is red.

In (9), "is" is best interpreted as something like $\text{has-color}(x,y)$, while in (10), "is" denotes equality, $\text{=}(x,y)$. Ideally, a CNL interpreter would make such determinations automatically. If it does this, then the interpretations of (9) and (10) will be the same although the surface syntactic forms differ. This is desirable, as (9) and (10) are essentially different ways of saying the same thing.

Form: But who says that $\text{has-color}(x,y)$ is the right interpretation of "is" in (9)? If the CNL interpreter makes this interpretation, then it forced an ontological commitment - that the ontology should include $\text{has-color}(x,y)$ - on the user. This seems undesirable to me. Again, the user, not the system, should decide on the ontology to use and be in control. In addition, (9) and (10) together violate the "one sense per word" principle, and thus is a potential recipe for confusion for the user.

Nat: But is that really a problem? It seems more of a problem to me to have lots of $\text{be}(x,y)$ predicates in the interpretation, all meaning different things.

Form: It is the user's job to be consistent about how they use a predicate like $\text{be}(x,y)$, and the CNL's job to be consistent in the interpretation.

Nat: That seems pretty constraining to me, when we're talking about ambiguous verbs like "be" or "have".

Form: Users are smart, they can work it out and be precise!

Nat: But why impose such constraints in the first place? There has been substantial progress in language processing technology over years that can be exploited, so that users can write more naturally.

Let's discuss another example of canonicalization, namely the interchangeable use of verbs and their nominalizations⁴. Consider:

(11) The army destroyed the city at night.

(12) The destruction of the city by the army was at night.

Again, as (11) and (12) mean essentially the same things. It would thus be nice if the CNL could produce the same canonical representation.

Form: Ugh, that is making the interpreter way too complex and unpredictable. Basically, the more fancy stuff the interpreter does, the less the user knows

⁴ See [19] for a good discussion of this topic.

what to expect. Better to have a more surface-level, direct mapping from the sentences to the logic that the user can get his head around.

Nat: What about actives and passives?

(13) A man loves a woman.

(14) A woman is loved by a man.

Should these produce different interpretations?

Form: Well, sure, those can be mapped to the same representation.

Nat: So some canonicalization is ok?

Form: Sure, if it's simple cases like that.

Nat: Where do you draw the line?

Form: Well, some minimal canonicalization like that is okay; the important thing is that it is pretty minimal and easy for the user to grasp.

[Use of a pre-defined ontology / ontological commitment]

Nat: I find it interesting that you insist on restricting/removing structural ambiguity, yet leave lexical interpretation completely unrestricted. Isn't that somewhat contradictory?

Form: Not at all. There is an important distinction between a language's *form* and *content*; a CNL should have unambiguous form, yet leave the content completely up to the user, if that's what he wants. Also, remember that although a "formalist" CNL doesn't dictate what words *mean* (that's up to the user), it *does* insist that words are unambiguous (one sense per word). I am perfectly consistent: A CNL should have unambiguous form but unrestricted content.

Also, there is nothing preventing the use of a pre-defined ontology, if one is available, with a "formalist" CNL. If you already have an axiomatized ontology, you can simply plug it in and have the user work with it, providing it obeys the "one sense per word" principle.

Rather, it seems to me that *you* might be the one being self-contradictory! You are happy with ambiguous language but then want to force a pre-defined ontology on the user. What kind of freedom is that?

Nat: Well, a naturalist CNL doesn't have to force a complete ontology on the user, it only needs the parts necessary to represent different disambiguation decisions (e.g., for representing the different meanings of "of"). So there is still flexibility. However, in many cases it's helpful, not harmful, to give the user a pre-defined ontology. Ontology-building is hard, and authoring disambiguation rules harder, and so if we can give both to the user, isn't that a good thing? Otherwise the user has to build it all from scratch!

Form: It seems to me that you are on a slippery slope: If you allow ambiguity then you need disambiguation rules, and disambiguation rules require world knowledge, and world knowledge requires an ontology to express it. So before you know it your CNL is ontology-specific. That's bad!

Nat: That's only partially true as the ontological commitment can be quite minimal, if we mainly use syntactic and lexical knowledge for disambiguation. But look, ontological commitment is generally a helpful thing! Otherwise the user is starting from scratch.

Form: Maybe he *wants* to start from scratch.

Nat: In which case he can use a "naturalist" CNL with the minimal pre-defined ontology.

Form: Or he could just use a "formalist" CNL and have complete freedom!

Nat: But have more work...!

Form: But have less confusion...!

[Practicality and Cost]

Form (continued): In any case, I remain skeptical it's even possible to build a decent "naturalist" CNL that's close to natural language - you are going to need a *lot* of heuristics to make these subtle disambiguation choices correctly. That's a lot of work! Full natural language processing has not made much headway in that direction.

Nat: Yes, but remember the language is constrained, making things a lot easier.

Form: I don't think that's enough. Disambiguation is highly domain- and word-specific; you are never going to be successful trying to build a domain-general, "naturalist" CNL. It would be prohibitively expensive!

Nat: Well, it's true that "naturalist" CNLs are probably more expensive to build. But that expense is for a good purpose, namely to make life easier for the users, as the machine is then doing some of the work for them. It seems like money well spent to me. Otherwise, the user will have to do this work himself, carefully crafting his wording to get the output he wants – there's no free lunch!

There are also several ways to potentially reduce cost: First, modern machine learning techniques can potentially help acquire disambiguation knowledge⁵, so it doesn't all have to be built by hand. Second, we could involve the users in entering the required knowledge via suitable knowledge acquisition tools. And third, remember we don't need "perfect" performance; as we agreed earlier, *any* CNL needs to present its interpretation for the user to check. This failure tolerance removes the need for perfection in the interpreter.

Form: But why try to make the software smart like this in the first place? I don't think users want smart software, they want controllable and predictable software. CNLs should be as controllable and predictable as a programming language, only more friendly to use.

Nat: No, I would say CNLs should be as natural and easy for the user as normal English, only constrained so that the computer can reliably find the intended interpretation.

⁵ e.g., [20,21].

[Handling genuine ambiguity, and combining the two]

Form: So what would a "naturalist" CNL do if a sentence is truly ambiguous? e.g.,

(15) A man sees a girl with a telescope.

(16) A lion is a wild animal.

(17) Three men have five cars.

Here you can't rely on heuristics to get the "right" interpretation!

Nat: Well, nothing different; the system makes its best guess and if it's wrong, the user can rephrase the sentence.

Form: Yes, but how does the user know how to rephrase if he doesn't fully understand the interpretation rules?

Nat: Well, the system could offer advice, or the user could rephrase in a less ambiguous way.

You know, there might, indeed, be a case for including some sentence patterns with clearly defined interpretations within the CNL for such eventualities.

Form: This sounds like adding in aspects of a "formalist" CNL!

Nat: Well, maybe some synthesis of the two approaches is the right way forward.

Form: Perhaps!

[Closing]

Nat: Look, I think our opinions originate from rather different philosophical perspectives: Should CNLs be more like an English-like specification language ("formalist"), or more like simplified natural language processing ("naturalist")? And in fact, in implementation terms, there is a continuum between the two extremes - a formalist-like CNL might still include some canonicalization, word-sense disambiguation, etc., or a naturalist-like CNL might include a deterministic core within it. Perhaps no CNL is in fact purely "formalist" or purely "naturalist", and these extremes may not even be fully definable or implementable. Ultimately, what really matters is the practical question: what do users find easiest to understand and work with?

Form: And that may be an empirical question to explore in the future.

Nat: Indeed. Shall we adjourn, and continue over a nice glass of Sicilian wine?

Form: That I can agree to!

Acknowledgements

We are grateful to Vulcan Inc. who supported part of this work through Project Halo, and to Norbert Fuchs for valuable discussions, comments, and organizing the CNL'2009 Workshop itself.

References

1. Clark, P., Harrison, P., Jenkins, T., Thompson, J., Wojcik, R.: Acquiring and Using World Knowledge using a Restricted Subset of English. In: Proc. FLAIRS 2005 (2005)
2. Pease, A., Murray, W.: An English to Logic Translator for Ontology-Based Knowledge Representation Languages. In: NL Processing and Knowledge Engineering (2003)
3. Fuchs, N.E., Schwertel, U., Schwitter, R.: Attempto Controlled English. In: Flener, P. (ed.) LOPSTR 1998. LNCS, vol. 1559, p. 1. Springer, Heidelberg (1999)
4. Schwitter, R.: English as a Formal Specification Language. In: Proc. 3rd, Proc. 13th Int. Workshop on Database and Expert Systems Applications (DEXA 2002): W04: Natural Language and Information Systems - NLIS, Aix-en-Provence, France, pp. 228–232 (2002)
5. Sowa, J.: Common Logic Controlled English. Technical Report (2004)
6. Tablan, V., et al.: User-friendly ontology authoring using a CL. In: ISWC 2008 (2008)
7. Clark, P., Chaw, J.: Capturing and Answering Questions Posed to a Knowledge-Based System. In: Proc. 4th Int. Conf. on Knowledge Capture, KCap 2007 (2007)
8. Barker, K., Porter, B., Clark, P.: A Library of Generic Concepts for Composing Knowledge Bases. In: Proc. 1st Int. Conf. on Knowledge Capture, KCap 2001 (2001)
9. Friedland, N., et al.: Project Halo: Towards a Digital Aristotle. *AI Magazine* 25(4) (2004)
10. Barker, K., Chaudhri, V., Chaw, S., Clark, P., Fan, J., Israel, D., Mishra, S., Porter, B., Romero, P., Tecuci, D., Yeh, P.: A Question-Answering System for AP Chemistry: Assessing KR&R Technologies. In: Proc. 9th International Conf. on Knowledge Representation and Reasoning, KR 2004 (2004)
11. Harrison, P., Maxwell, M.: A New Implementation of GPSG. In: Proc. 6th Canadian Conf. on AI (1986)
12. Clark, P., Harrison, P.: BLUE (Boeing Language Understanding Engine): A Quick Tutorial on How it Works. Working Note 35 (2009),
http://www.cs.utexas.edu/users/pclark/working_notes/
13. Barker, K., et al.: Learning by Reading: A Prototype System. In: Proc. AAAI 2007 (2007)
14. Chaudhri, V., Barker, K., Clark, P.: AURA: Automated User-Centered Reasoning and Acquisition System: Phase II Final Report. Technical Report ICS-17540-FR-09, SRI International (2009)
15. Kaljurand, K.: Paraphrasing controlled English texts, In: Proc. CNL 2009 (CEUR Workshop Proceedings), vol. 448 (2009)
16. Tennant, H., Ross, K., Saenz, R., Thompson, C., Miller, J.: Menu-Based Natural Language Understanding. In: Proc. SIGCHI Conf. on Human Factors in Computing Systems, pp. 154–160 (1983)
17. Kuhn, T., Schwitter, R.: Writing Support for Controlled Natural Languages. In: Powers, D., Stockes, N. (eds.) Proceedings of ALTA 2008, pp. 46–54 (2008)
18. Schwitter, R., Ljungberg, A., Hood, D.: ECOLE – A Look-ahead Editor for a Controlled Language. In: Proceedings of EAMT-CLAW 2003, pp. 141–150 (2003)
19. Gurevich, O., Crouch, R., King, T., de Paiva, V.: Deverbal Nouns in Knowledge Representation. In: Proc. FLAIRS 2006 (2006)
20. Gale, W., Church, K., Yarowsky, D.: A Method for Disambiguating Word Senses in a Large Corpus. *Computers and the Humanities* 26, 415–439 (1992)
21. Toutanova, K., Haghighi, A., Manning, C.: A Global Joint Model for Semantic Role Labeling. *Computational Linguistics* 34(2), 161–191 (2008)

Implementing Controlled Languages in GF

Krasimir Angelov and Aarne Ranta

Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract. This paper introduces GF, Grammatical Framework, as a tool for implementing controlled languages. GF provides a high-level grammar formalism and a resource grammar library that make it easy to write grammars that cover similar fragments in several natural languages at the same time. Authoring help tools and automatic translation are provided for all grammars. As an example, a grammar of Attempto Controlled English is implemented and then ported to Finnish, French, German, Italian and Swedish.

1 Introduction

The term *controlled language* is used in many ways, ranging from formalism-like systems with strict syntax and semantics such as Rabbit [1] to rich fragments of natural language informally specified by heuristic rules such as Boeing Simplified English [2]. The former kind is readily usable for mechanical processing, while the latter kind is designed for human users. In this paper, we do not aim to settle the question about the true nature of controlled languages, let alone suggest yet another such language. The goal is instead to present an *implementation framework*, which can cope with the rich variety of existing controlled languages and also boost the development of new ones. To demonstrate the adequateness of the framework, we have reverse-engineered one representative controlled language, ACE (*Attempto Controlled English*) [3]. To show the value added by the new implementation, we have furthermore ported ACE from English to five other languages: Finnish, French, German, Italian, and Swedish. This additional work shows one of our framework's main strengths, *multilinguality*, which readily enables one of the most promising potential applications of controlled languages: meaning-preserving high-quality automatic translation.

According to any of the diverging notions, a controlled language mimics some existing natural language—most frequently English—but has a restricted syntax and semantics. The aim of the restrictedness is to reduce ambiguity and help human understanding, and often also enable mechanical processing. The advantage of such a *sublanguage* of a human language over purely formal languages is that humans can understand it without special training, provided they know the *superlanguage*. Yet a controlled language need not be strictly a part of its superlanguage; it can have additional constructs that resolve ambiguities, or change some grammar rules by ignoring inflection, etc.

As *ambiguity* is one of the main problems in the mechanical processing of human language, most controlled languages are specially designed in a way that reduces ambiguity. Sometimes, however, ambiguity does not really matter. For example, structural ambiguities such as quantifier scope do not harm in machine translation, if the same

ambiguity is present in both the source and the target language. On the other hand, an ambiguity in a voice-control system can lead to a catastrophically wrong reaction from the system. Such ambiguities must be resolved either by the design of the language or by using clarification questions at run time. In general, ambiguity is a matter of trade-off: more strict languages might sound less natural but are easier to process, whereas more ambiguous languages might be more natural but harder to process.

A common application of controlled languages is *technical documentation*, where the use of controlled language aims to ensure that the documentation will be interpreted correctly by the reader. The ultimate source of such documentation can be a formalized knowledge base, from which the document is automatically generated, but it can also be a human author trained to conform to the rules of the controlled language. In the latter case, no underlying formal model is required.

Controlled languages that do have formal semantic models are often connected with *reasoning engines*. The controlled language may be used as an input format, so that its utterances are converted to logical formulas or database formalism, which the reasoning engine manipulates. The resulting answer may then be translated back to the controlled language to the benefit of the human user.

2 The Framework Idea

The multitude of purposes, semantic models, and application domains suggests that it is not feasible to build one controlled language to be used for all purposes: different applications need different language fragments, and unnecessary unions of such fragments would even give rise to unexpected ambiguities. Thus it seems desirable to tailor special languages to different purposes. But defining a new language and implementing its processing and authoring tools is a considerable effort, and one would not like to repeat this effort for each new application.

The grammar formalism GF (Grammatical Framework) [4] is a tool for defining and implementing languages. It started as an experimental system at Xerox [5] and has later been used in numerous natural language applications, most of which can be viewed as a domain-specific controlled languages: mathematical proof editing [6], software specification authoring [7], dialog system for in-car music players [8], dialog system for route finding [9], translation of mathematical exercises [10], multilingual wiki for restaurant reviews [11], and multilingual generation in the cultural heritage domain [12]. The forthcoming European project MOLTO (Multilingual On-Line Translation, www.molto-project.eu) aims to boost the usability of GF to new and more comprehensive applications.

The language fragment used in mathematics is obviously different from the fragment used in restaurant reviews. Such fragments can differ not only in the choice of the vocabulary or in the pragmatic use of the language but also in the semantics. A general-purpose controlled language might choose to use Montague-style semantics [13] in a logical calculus, to cover many different domains at the same time. The final output from the semantic analyses is then a logical formula, which can be processed by theorem provers and similar tools. But this is not necessarily the best choice

for every application. For example, in a dialogue system for a travel agency, the semantic representation of the sentence *I want to fly from Paris to Rome* should be no more complicated than

(*Fly Paris Rome*)

This is all the information that is needed for the rest of the system to process the sentence. A general-purpose logical semantics would get entangled in irrelevant details, such as the meanings of the words *I* and *want*.

Controlled language design has a parallel in the foundations of mathematics. One approach is monolithic foundations, where everything is reduced to one formal system such as Zermelo-Fraenkel set theory. This approach is contrasted by the idea of *logical frameworks* [14,15], where the common foundation is not a particular theory or logic but just a framework for defining logics; the framework accommodates many different logics, and does not try to decide about one ultimate correct logic.

Another related idea is Wittgenstein's philosophical notion of *language games*. Language as a whole does not have a strictly formal structure and cannot be formally defined. Nevertheless, there are small sublanguages that are used in specific situations "for doing things", inseparably from their contexts of use. These sublanguages are more predictable and well defined than language as a whole, and one can often describe them by precise formal rules.

Yet another parallel is the modern movement in *ontologies*. The development has shifted from building large universal ontologies to small domain specific ones; OWL (Web Ontology Language, [16]) is a framework for such ontologies. Small ontologies are easier to design because the domain is better defined. The design of a universal ontology is a never-ending task, and it would be unpractical to use even if it were theoretically possible. In any particular application, large parts of a universal ontology would remain unused, and the relevant parts would hardly be precise enough for the domain in question.

3 GF and Multilingual Grammars

GF uses a logical framework based on Martin-Löf's type theory [15] for building semantic models of languages. To the logical framework, GF adds a syntax formalism that defines the realizations of formal meanings as concrete linguistic expressions. The semantic model part is in GF called *abstract syntax*, following standard computer science terminology; the syntactic realization part is called *concrete syntax*.

As a first example, the abstract syntax in Figure 1 gives a model of simple mathematical exercises. It declares the three *categories* of problems, propositions, and expressions, as well as six *functions* for constructing *abstract syntax trees*. A tree in the abstract syntax is a Lisp-like lambda term: a function applied to zero or more trees. For example,

Prove (Even (Sqrt (EInt 36)))

is a tree representing the problem statement *prove that the square root of 36 is even*. Unlike in Lisp, GF trees are statically typed. The type checker can verify that the above tree is a well-typed one in the category *Problem*.

```

abstract Math = {
  cat Problem; Prop; Exp;
  fun Prove : Prop → Problem;
    Compute : Exp → Problem;
    Even : Exp → Prop;
    Div : Exp → Exp → Prop;
    Sqrt : Exp → Exp;
    EInt : Int → Exp;
}

```

Fig. 1. An abstract syntax for mathematical problems

```

concrete MathEng of Math = {
  lincat Problem, Prop, Exp = Str;
  lin Prove p = "prove that" ++ p;
    Compute e = "compute" ++ e;
    Even x = x ++ "is even";
    Div x y = x ++ "is divisible by" ++ y;
    Sqrt x = "the square root of" ++ x;
    EInt x = x;
}

```

Fig. 2. An English concrete syntax for mathematical problems

A concrete syntax defines, for each abstract syntax category, its *linearization type*, and for each function, its *linearization function*. A simple concrete syntax for *Math* is shown in Figure 2. In this grammar, all categories have *Str* (strings; more accurately token lists) as their linearization type, and all linearization functions are ones that convert trees to strings. The operator `++` concatenates token lists.

For any given abstract-concrete syntax pair, GF implements its core functionalities (as well as many other ones):

- parsing: a mapping from strings to trees
- linearization: a mapping from trees to strings
- type checking of trees
- random generation of trees
- semantic evaluation and paraphrasing
- authoring help: syntax editing and word completion

Due to its design distinguishing abstract from concrete syntax, GF is *multilingual*: one abstract syntax can be given several concrete syntaxes. Thus *translation* is obtained by combining parsing and linearization: the source language sentence is first parsed to an abstract syntax tree and then linearized to a sentence in the target language. As the abstract syntax reflects the semantics of the domain instead of the syntax of some concrete language, the translation is meaning-preserving by design.

```

concrete MathFre of Math = {
  lincat Problem, Prop, Exp = Str;
  lin Prove p = "démontrer que" ++ p;
    Compute e = "calculer" ++ e;
    Even x = x ++ "est pair";
    Div x y = x ++ "est divisible par" ++ y;
    Sqrt x = "la racine carrée de" ++ x;
    EInt x = x;
}

```

Fig. 3. A naïve French concrete syntax for mathematical problems

Figure 3 illustrates multilinguality with a naïve concrete syntax for French. We can now linearize our example tree to *démontrer que la racine carrée de 36 est pair*, to obtain a translation of *prove that the square root of 36 is even*. This example shows immediately that the French grammar is not correct: it does not support gender agreement, which dictates that even though *2 est pair* is correct, a feminine subject requires another form of the adjective: *la racine carrée de 36 est **paire***.

In a larger fragment, we would likewise see that the grammar does not make mood distinctions (*2 est pair* vs. *il existe un entier x tel que x **soit** pair*), nor does it implement preposition-article contractions (*la racine carrée d'un entier*, *la racine carrée **du** successeur de x*). Such problems can be fixed by introducing *parameters* in the concrete syntax in Figure 4. Parameters like gender and mood work very much like features in unification grammars [17]. But in GF, they are strictly a part of a concrete syntax and thus neatly separated from the semantic description of the domain; different languages may employ different sets of parameters for the same abstract syntax.

```

concrete MathFre of Math = {
  param Mood = Ind | Subj;
    Gender = Masc | Fem;
  lincat Prop = Mood  $\Rightarrow$  Str;
    Exp = {s : Str; g : Gender};
  lin Even x = table {
    Ind  $\Rightarrow$  x.s ++ "est" ++ pair;
    Subj  $\Rightarrow$  x.s ++ "soit" ++ pair
  } where {
    pair = case x.g of {Masc  $\Rightarrow$  "pair"; Fem  $\Rightarrow$  "paire"}
  };
}

```

Fig. 4. A corrected French concrete syntax for mathematical problems (fragment)

4 Resource Grammar Libraries

Although the mathematical example above is very simple, it is complex enough to show that a lot of linguistic knowledge is needed for defining a concrete syntax. This is an important management problem because it means that the grammar developer has to be both an expert in linguistics and a domain expert. The solution in GF, suggested by other fields of software engineering, is to use *libraries* [18].

The *GF Resource Grammar Library* [19] is a collection of wide coverage grammars, which are currently available for fifteen natural languages: Bulgarian, Catalan, Danish, Dutch, English, Finnish, French, German, Italian, Norwegian, Polish, Romanian, Russian, Spanish, and Swedish. The libraries are based on linguistic concepts and hence independent of domain-specific semantics. They are implemented by linguistic experts but designed to be used by nonlinguists. The libraries enable a division of labour between linguists and domain experts and eliminates the repetition of linguistic work for every new domain.

Just like application grammars, the resource grammar library is based on a common abstract syntax. This abstract syntax can be seen as a “linguistic ontology”, which contains syntactic structures that can be found in different languages. One example is the predication function,

$$\text{fun PredVP} : NP \rightarrow VP \rightarrow CI$$

which combines noun phrases and verb phrases into clauses. The exact details of predication (agreement, word order, tenses, moods) vary greatly from one language to another, but the user of the library can be sure that these details are correctly defined in the concrete syntaxes of the library. The user only needs to see the abstract syntax, i.e. what functions are available.

To make the library even easier to use, its API (Application Programmer’s Interface) is presented as a set of *overloaded* functions. These functions permit a memorizable naming convention: all functions whose value is category *C* has the name *mkC*. For instance, *PredVP* is accessible as *mkCI*. As the interpretation of overloading is based on type checking, overloading is possible for any set of functions that have different types (see [18]). For other functions, the library API still uses names suffixed by the value category. As an example, Figure 5 shows a correct, scalable implementation of the French example grammar using the GF resource grammar library. The library module *SyntaxL* defines the syntactic structures of each language *L*, whereas *ParadigmsL* defines its inflection paradigms. Depending on *L*, these paradigms need different pieces of information to derive the full inflection of words¹.

The English implementation would be very similar. The main difference is in words: *démontrer* is replaced by *prove*, *pair* by *even*, and so on. But the syntactic structure, i.e. the combinations of resource grammar API functions, is the same, with one exception: the main verb in problem statements, which in French is just the infinitive, becomes imperative in English:

$$\text{Prove } p = \text{mkUtt } (\text{mkImp } (\text{mkVP } (\text{mkVS } \text{"prove"}) (\text{mkS } p)));$$

and similarly for *Compute*.

¹ For the library API, see grammaticalframework.org/lib.


```

concrete MathFre of Math = open SyntaxFre, ParadigmsFre in {
  lincat Problem = Utt;
    Prop = Cl;
    Exp = NP;
  lin Prove p = mkUtt (mkVP (mkVS "démontrer") (mkS p));
    Compute e = mkUtt (mkVP (mkV2 "calculer") e);
    Even x = mkCl x (mkA "pair");
    Div x y = mkCl x (mkA2 "divisible" "par") y;
    Sqrt x = mkNP defArt (mkCN (mkCN (mkN "racine")
                                   (mkA "carré" (mkAdv de_Prep x))));
    EInt x = mkNP x;
}

```

Fig. 5. A library-based French concrete syntax for mathematical problems

Multilingual grammars in GF support semantics-preserving translation. Linearization rules that use the same API functions can moreover be said to be *syntax-preserving*. Even though semantic-preserving translation does not need to be syntax-preserving, experience in GF projects shows that it mostly is—perhaps in 90% of cases. This means that it makes sense to use *functors* to define concrete syntaxes of several languages at the same time. A functor is a concrete syntax module that is parametrized on an *interface*—a set of parameters, which are implemented separately for each language. For instance, the interface for mathematical exercises has constants such as

```

prove_VS : VS;
even_A : A;
mkProblem : VP → Utt;

```

Except for *mkProblem*, the parameters are for individual words. The French definitions of these parameters are

```

prove_VS = mkVS "démontrer";
even_A = mkA "pair";
mkProblem = mkUtt;

```

whereas the English definitions are

```

prove_VS = mkVS "prove";
even_A = mkA "even";
mkProblemp = mkUtt (mkImp p);

```

Technically, the resource grammar API *Syntax* is just another interface. A typical GF application is thus a functor over two interfaces: the resource API and a domain-dependent lexicon with a few syntactic functions. The work required when porting the application to a new language is just to write a new instance of the domain interface.

5 Attempto in GF

To give a full-scale demonstration of the use of GF in implementing controlled languages, we have written a GF grammar for Attempto Controlled English (ACE) [3]. ACE is a general-purpose controlled language with fixed syntax but with a lexicon that is customized for every application. There is a complete translation from ACE into first-order logic and a partial mapping into OWL [16]; not every sentence in ACE is expressible as an OWL statement. ACE is also accompanied with tools for generation, parsing, and authoring.

We first implemented English as specified in the ACE documentation, and then ported it to Finnish, French, German, Italian, and Swedish. Thus we can now translate Attempto documents between all these languages, and use these languages to give input to Attempto applications. We can also combine GF authoring tools with Attempto tools; since we can let Attempto take care of, for instance, anaphora resolution and logical reasoning, there is no need to do rebuild them in GF. By using English as input language, these tools become available in the other languages as well.

ACE is originally defined by a unification grammar. Our implementation in GF followed the document *ACE 6.0 Construction Rules* [3]. The more formal document *ACE 6.0 Syntax Report* [3] was used as a reference, but we did not follow it in detail, since it follows English too closely to be generalizable in the way we wanted.

The implementation steps followed a pattern familiar from previous work, e.g. [8]. Parts of the modules built are shown in Appendix A. These parts cover about 20% of the whole grammar, defining a fragment that is sufficient for most examples in this paper.

- First, refactor the Attempto grammar into an abstract syntax *Attempto* (A.1) and a concrete syntax *AttemptoEng*, and implement the latter by using the GF resource library for English.
- Second, refactor the English concrete syntax into a functor *AttemptoI* (A.2) and a domain interface *LexAttempto* (A.3), and implement the latter as *LexAttemptoEng* (A.4). *AttemptoEng* now becomes a simple functor instantiation, which in addition to the domain interface uses the resource grammar interfaces *Syntax* (syntactic structures) and *Symbolic* (embedding symbolism in text) (A.5).
- Third, write *LexAttemptoGer* (A.4) and (mechanically), *AttemptoGer* (A.5), to obtain a German version of the Attempto language. This step can then repeated for French, Swedish, Italian, Finnish—and any other language supported by the GF Resource Grammar Library.

The resulting module structure, displayed in Figure 6, has become something like a “design pattern” for GF grammars. On top of the core structure, one can build a domain lexicon (*TestAttempto* in our experiment), which can of course have its own internal structure. In the figure, rectangles mark abstract syntax and ellipses mark modules needed in concrete syntax. The dashed ellipses are the only ones that require human work when the grammar is ported to a new language.

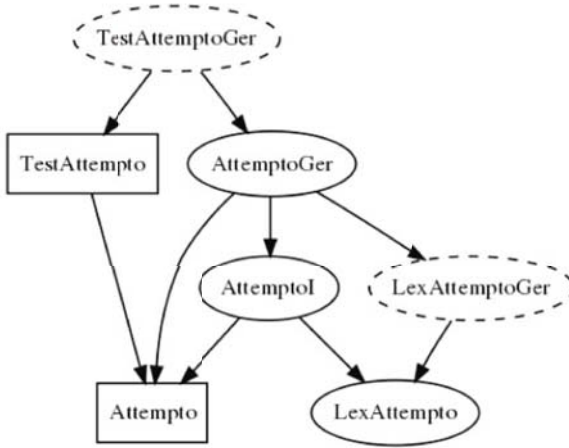


Fig. 6. The dependency graph of the modules in the ACE grammar

As an example of the Attempto test lexicon, the abstract syntax constant *customer_N* is linearized as follows:

English:	<i>mkN</i> "customer"
Finnish:	<i>mkN</i> "asiakas"
French:	<i>mkN</i> "client"
German:	<i>mkN</i> "Kunde" "Kunden" <i>masculine</i>
Italian:	<i>mkN</i> "cliente"
Swedish:	<i>mkN</i> "kund" "kunder"

Thus the Attempto tree

vpS (everyNP (adjCN (positAP rich_A) customer_N)) (apVP (positAP important_A))

has the following linearizations in the six languages:

English:	<i>every rich customer is important</i>
Finnish:	<i>jokainen rikas asiakas on tärkeä</i>
French:	<i>chaque client riche est important</i>
German:	<i>jeder reiche Kunde ist wichtig</i>
Italian:	<i>ogni cliente ricco è importante</i>
Swedish:	<i>varje rik kund är viktig</i>

6 Results

In rough figures, the Attempto grammar has about 30 categories, 160 syntax rules, and a test lexicon of 80 words. It took about one day to design the abstract syntax and two days

to write the concrete syntax for the first language (English). One more day was needed to split the grammar into a functor and an interface, with an English implementation. After that it was a matter of hours to instantiate the functor to each new language.

Writing the GF implementation was mostly smooth. The main problem was that some ACE structures were not covered by the library, in particular those that are not quite correct in standard English. One example is negated noun phrases, especially in the object position: *John meets not every customer*.

As the grammar was ported to other languages by using a functor obtained from English, the result has sometimes a flavour of English syntax, even though grammatically correct. But it can get outright strange for constructs like *not every*. For instance, in French they create negative elements such as *pas* and *aucun* without the grammatically obligatory *ne*. With some more work, such problems could be solved by making the grammar less like Attempto English than we did—either by generalizing the functor, or even by changing the abstract syntax in some places. On the other hand, since Attempto French (as well as Finnish, German, Italian, Swedish) is an artificial language anyway, artificial modes of expression might well be tolerable in it, as long as it can be understood without training.

7 Authoring Tools

Many controlled language systems provide some kind of authoring tools, which guide the user to stay within the scope of the grammar. In general they should be as intuitive as possible and should resemble a familiar text editing environment. The characteristic of the controlled language, that it has clearly defined syntax and semantics, can be exploited by the environment to detect misspellings and syntactic and semantic errors with a high precision. This is in contrast to the traditional office products for free text where the grammar and spelling checkers are only approximate.

There are currently three different authoring tools in use with GF. Figure 7 shows a translator client using a GF web server. The client uses a *completion parser*, which here shows the grammatically correct completions of the letter *B* in the middle of a German sentence. Usually tools like this work, if not with a bare lexicon, with a context-free grammar that only gives a rough approximation of grammaticality. Even if context-free grammars can somehow be made to work for English, they become impossible or unpractical for languages with more complicated structure.



Fig. 7. Word Completion

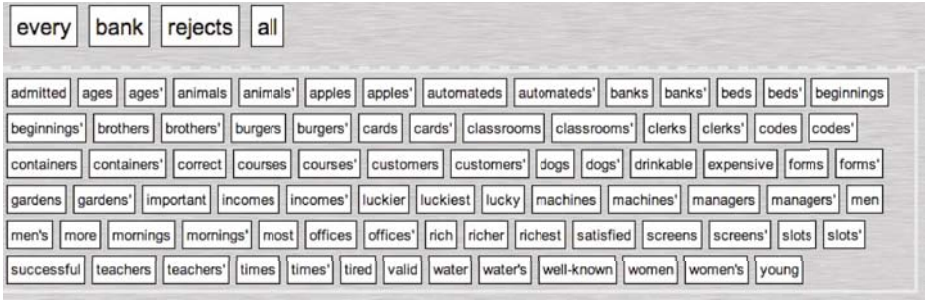


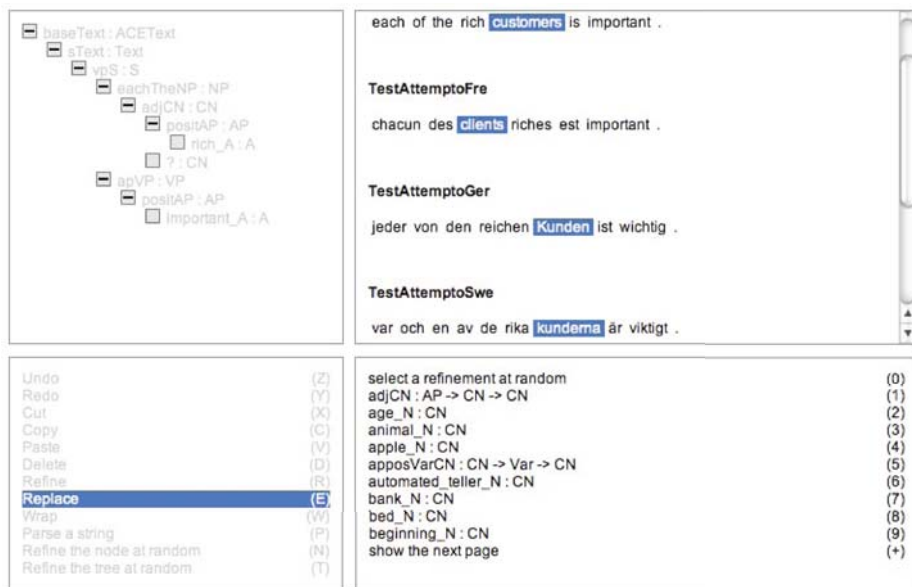
Fig. 8. Fridge Magnets

When the a GF grammar is compiled, it is transformed into a low-level format of *Parallel Multiple Context-Free Grammars* (PMCFG, [20]), as explained in [21]. A multilingual GF grammar becomes a *synchronous* PMCFG, which is analogous to the simpler (mostly context-free) synchronous grammars that are becoming more and more popular in machine translation (cf. [22], [23]). The usage of a more expressive grammar formalism ensures that even complex language constructs can be handled adequately. PMCFG as such is a very low-level formalism, which would be hard to write by hand. But in GF, it is generated via compilation from GF source code, which has the right level of abstraction and is human-readable.

The parser that we use [24] works with PMCFG and can successfully handle long distance relations and discontinuous phrases. The parser is incremental, which is what enables authoring tools like the one in Figure 7. The parser reads the already written tokens and constructs the forest of trees that are consistent with the current input. Using the forest, it computes the set of tokens that are allowed to follow the already written ones. The set is moreover filtered by the prefix of the word if there is one, e.g. the letter *B* in Figure 7.

Another user interface exploiting incremental PMCFG parsing has the form factor of Fridge Magnets (Figure 8). In this interface, words are shown as little boxes (magnets) that can stick to the fridge door (the grey background). The interface emulates the fridge magnets game where the user can compose sentences on the fridge door by combining magnets. The real game with a static set of physical magnets cannot guarantee the grammaticality of the sentences formed, and usually doesn't even provide all inflectional forms needed. But the virtual GF version shows all and only the words that complement the partial sentence so that the phrase stays grammatical and can eventually be completed to a full sentence.

A different interface technique is used in the *syntax editor* (Figure 9), where the user manipulates the syntax tree directly rather than writing sentences [25]. This interface is suitable when the user has to modify an already existing document and the modification involves non-local changes. For example, replacing the noun *customer* by *bank* in Figure 9 results in French with the sentence *chacune des banques riches est importante*, where also the first and the last words are changed because of gender agreement. In a parsing-based interface, such implied changes must be made manually, while the syntax editor performs them automatically. On the other hand, the syntax editor may not be the



tool of choice when writing a new text from scratch, since its mode of operation is very unfamiliar to most users.

Since the parsing-based and syntax-tree-based editors have different advantages in different uses, the best system is probably a combination of them, a “pluralistic editor” to use a term from [26]. The GF incremental parser and syntax editor functionalities are available as components that can be combined into such systems [27].

8 Related Work

In addition to type theory, GF was inspired by the WYSIWYM system (What You See is What You Mean, [28]), which is a multilingual authoring system for controlled languages based on a common semantics. The resource grammar library was inspired by CLE (Core Language Engine, [29]), which uses a multilingual general-purpose grammar specializable to domains of application.

9 Conclusion

GF is a programming language for multilingual grammars with shared abstract syntaxes. Any multilingual grammar defined in GF is equipped with a parser, a generator, a translator, and several authoring interfaces. GF has been used in applications ranging from mathematical proof systems to spoken dialogue systems. An experiment with Attempto Controlled English shows how GF can be used for implementing controlled languages and generalizing them to multilingual systems, to create semantic-based systems for translation and multilingual documentation.

References

1. Hart, G., Johnson, M., Dolbear, C.: Rabbit: Developing a control natural language for authoring ontologies. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) *ESWC 2008*. LNCS, vol. 5021, pp. 348–360. Springer, Heidelberg (2008)
2. The Boeing Company: Boeing Simplified English Checker (2001), <http://www.boeing.com/assocproducts/sechecker>
3. Fuchs, N.: Attempto project homepage (2008), <http://attempto.ifi.uzh.ch/site/>
4. Ranta, A.: Grammatical Framework: A Type-Theoretical Grammar Formalism. *Journal of Functional Programming* 14(2), 145–189 (2004)
5. Dymetman, M., Lux, V., Ranta, A.: XML and multilingual document authoring: Convergent trends. In: *COLING*, Saarbrücken, Germany, pp. 243–249 (2000)
6. Hallgren, T., Ranta, A.: An extensible proof text editor. In: Parigot, M., Voronkov, A. (eds.) *LPAR 2000*. LNCS (LNAI), vol. 1955, pp. 70–84. Springer, Heidelberg (2000)
7. Burke, D.A., Johannisson, K.: Translating Formal Software Specifications to Natural Language / A Grammar-Based Approach. In: Blache, P., Stabler, E., Busquets, J., Moot, R. (eds.) *LACL 2005*. LNCS (LNAI), vol. 3492, pp. 51–66. Springer, Heidelberg (2005)
8. Perera, N., Ranta, A.: Dialogue System Localization with the GF Resource Grammar Library. In: *SPEECHGRAM 2007: ACL Workshop on Grammar-Based Approaches to Spoken Language Processing*, Prague, June 29 (2007)
9. Bringert, B., Cooper, R., Ljunglöf, P., Ranta, A.: Multimodal dialogue system grammars. In: *Proceedings of DIALOR 2005, Ninth Workshop on the Semantics and Pragmatics of Dialogue*, June 2005, pp. 53–60 (2005)
10. Caprotti, O.: WebALT! Deliver Mathematics Everywhere. In: *Proceedings of SITE 2006*, Orlando, March 20–24 (2006)
11. Meza Moreno, M.S., Bringert, B.: Interactive multilingual web applications with grammatical framework. In: Nordström, B., Ranta, A. (eds.) *GoTAL 2008*. LNCS (LNAI), vol. 5221, pp. 336–347. Springer, Heidelberg (2008)
12. Dannélls, D.: Generating Tailored Texts for Museum Exhibits. In: *Proceedings of the 6th edition of LREC. The 2nd Workshop on Language Technology for Cultural Heritage (LaTeCH 2008)*, Marrakech, Morocco, pp. 17–20 (2008), <http://spraakdata.gu.se/svedd/pub/latech08.pdf>
13. Montague, R.: *Formal Philosophy*. Yale University Press, New Haven (1974); Collected papers edited by Richmond Thomason
14. Harper, R., Honsell, F., Plotkin, G.: A Framework for Defining Logics. *JACM* 40(1), 143–184 (1993)
15. Nordström, B., Petersson, K., Smith, J.: *Programming in Martin-Löf's Type Theory. An Introduction*. Clarendon Press, Oxford (1990)
16. Dean, M., Schreiber, G.: *OWL Web Ontology Language Reference* (2004), <http://www.w3.org/TR/owl-ref/>
17. Shieber, S.: *An Introduction to Unification-Based Approaches to Grammars*. University of Chicago Press, Chicago (1986)
18. Ranta, A.: Grammars as Software Libraries. In: Bertot, Y., Huet, G., Lévy, J.-J., Plotkin, G. (eds.) *From Semantics to Computer Science*, pp. 281–308. Cambridge University Press, Cambridge (2009)
19. Ranta, A.: The GF Resource Grammar Library. *Linguistic Issues in Language Technology* 2 (2009)
20. Seki, H., Matsumura, T., Fujii, M., Kasami, T.: On multiple context-free grammars. *Theoretical Computer Science* 88, 191–229 (1991)
21. Angelov, K., Bringert, B., Ranta, A.: PGF: A Portable Run-Time Format for Type-Theoretical Grammars. *Journal of Logic, Language and Information* (2009) (to appear)

22. Chiang, D.: A hierarchical phrase-based model for statistical machine translation. In: ACL 2005: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics, Morristown, NJ, USA. Association for Computational Linguistics, pp. 263–270 (2005)
23. Zollmann, A., Venugopal, A., Och, F.J., Ponte, J.: A systematic comparison of phrase-based, hierarchical and syntax-augmented statistical MT. In: Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008), August 2008, pp. 1145–1152 (2008)
24. Angelov, K.: Incremental parsing with parallel multiple context-free grammars. In: European Chapter of the Association for Computational Linguistics (2009)
25. Khegai, J., Nordström, B., Ranta, A.: Multilingual Syntax Editing in GF. In: Gelbukh, A. (ed.) CICLing 2003. LNCS, vol. 2588, pp. 453–464. Springer, Heidelberg (2003)
26. Welsh, J., Broom, B., Kiong, D.: A design rationale for a language-based editor. *Software–Practice and Experience* 21, 923–948 (1991)
27. Bringert, B., Angelov, K., Ranta, A.: Grammatical Framework Web Service. In: System demo, Proceedings of EACL 2009, Athens (2009)
28. Power, R., Scott, D.: Multilingual authoring using feedback texts. In: COLING-ACL (1998)
29. Alshawi, H.: *The Core Language Engine*. MIT Press, Cambridge (1992)

A The GF Code for Attempto

This Appendix shows the code needed for implementing a self-contained fragment of Attempto for English, French, and German using GF and its resource grammar library. The abstract syntax function declarations in Section A.1 show English and French examples as comments. The numeric references (like 2.1.1) in Section A.1 follow the *ACE 6.0 Construction Rules* in attempto.ifi.uzh.ch/site/docs. The GF library functions used in the grammar are documented in the resource grammar library synopsis in grammaticalframework.org/lib. The complete Attempto code for six languages can be found in grammaticalframework.org/examples/attempto. GF itself is open-source software available from grammaticalframework.org.

A.1 Attempto Abstract Syntax

```
abstract Attempto = {
cat
  S ;          -- sentence
  CN ;         -- common noun
  MCN ;        -- mass noun
  NP ;         -- noun phrase
  A ;          -- adjective
  AP ;         -- adjectival phrase
  RS ;         -- relative clause
  Pron ;       -- pronoun
  VP ;         -- verb phrase
  V ;          -- verb
  VS ;         -- sentence-complement verb
  V2 ;         -- noun phrase complement verb
  Conj ;       -- conjunction
  RP ;         -- relative pronoun
  Formula ;    -- symbolic formula
```



```

Term ;      -- symbolic term
Var ;      -- variable symbol
fun
-- 2.1.1
  aNP : CN -> NP ; -- a dog ; un chien
  theNP : CN -> NP ; -- the dog ; le chien
  everyNP : CN -> NP ; -- every dog ; chaque chien
-- 2.1.2
  theCollNP : CN -> NP ; -- the dogs ; les chiens
  allCollNP : CN -> NP ; -- all dogs ; tous les chiens
-- 2.1.3
  someMassNP : MCN -> NP ; -- some water ; quelqu'eau
  notAllMassNP : MCN -> NP ; -- not all water ; pas toute l'eau
-- 2.1.9
  he_NP : NP ; -- he ; il
  she_NP : NP ; -- she ; elle
-- 2.1.11
  somebody_NP : NP ; -- someone ; quelqu'un
-- 2.1.13
  nothing_butNP : CN -> NP ; -- nothing except dogs ;
                                -- rien excepté des chiens
-- 2.1.16
  lt_Formula : Term -> Term -> Formula ; -- x < y
  varTerm : Var -> Term ; -- x
  strVar : String -> Var ; -- x
  termNP : Term -> NP ; -- x + y
-- 2.2.1
  adjCN : AP -> CN -> CN ; -- rich customer ; client riche
  positAP : A -> AP ; -- rich ; riche
-- 2.2.2
  relCN : CN -> RS -> CN ; -- man who sleeps ; homme qui dort
  predRS : RP -> VP -> RS ; -- who sleeps ; qui dort
  slashRS : RP -> NP -> V2 -> RS ; -- that he eats ; qu'il mange
  which_RP : RP ; -- which ; que
-- 2.3.1
  vpS : NP -> VP -> S ; -- he sleeps ; il dort
  neg_vpS : NP -> VP -> S ; -- he doesn't sleep ; il ne dort pas
  vVP : V -> VP ; -- sleep ; dormir
  vsVP : VS -> S -> VP ; -- believe that he sleeps ;
                                -- croire qu'il dort
  v2VP : V2 -> NP -> VP ; -- fill in the form ;
                                -- remplir le formulaire
-- 2.3.2
  apVP : AP -> VP ; -- be rich ; etre riche
  compVP : A -> NP -> VP ; -- he is richer than she ;
                                -- il est plus riche qu'elle
-- 2.3.5
  canVP : VP -> VP ; -- can sleep ; pouvoir dormir
-- 3.2
  thereNP : NP -> S ; -- there is a bank ; il y a une banque

```

```

-- 3.3
  formulaS : Formula -> S ; -- x > y
-- 3.4.1
  coordS : Conj -> S -> S -> S ; -- he sleeps and she is tired ;
                                -- il dort et elle est fatiguée
  and_Conj : Conj ; -- and ; et
  or_Conj : Conj ; -- or ; ou
-- 3.4.4
  if_thenS : S -> S -> S ; -- if A then B ; si A alors B
  possibles : S -> S ; -- it is possible that A ;
                    -- il est possible que A
  necessaryS : S -> S ; -- it is necessary that A ;
                    -- il est nécessaire que A
}

```

A.2 The Concrete Syntax Functor

```

incomplete concrete AttemptoI of Attempto = open
  Syntax, Symbolic, Prelude, LexAttempto in {
lincat
  S = Syntax.S ;
  CN = Syntax.CN ;
  MCN = Syntax.CN ;
  NP = Syntax.NP ;
  A = Syntax.A ;
  AP = Syntax.AP ;
  RS = Syntax.RS ;
  Pron = Syntax.Pron ;
  VP = Syntax.VP ;
  V = Syntax.V ;
  VS = Syntax.VS ;
  V2 = Syntax.V2 ;
  Conj = Syntax.Conj ;
  RP = Syntax.RP ;
  Var, Term, Formula = {s : Str} ;
lin
  aNP = mkNP a_Art ;
  theNP = mkNP the_Art ;
  everyNP = mkNP every_Det ;
  theCollNP = mkNP the_Art plNum ;
  allMassNP cn = mkNP all_Predet (mkNP cn) ;
  someMassNP = mkNP someSg_Det ;
  notAllMassNP cn = mkNP not_Predet (mkNP all_Predet (mkNP cn)) ;
  he_NP = mkNP he_Pron ;
  she_NP = mkNP she_Pron ;
  somebody_NP = Syntax.somebody_NP ;
  nothing_butNP cn =
    mkNP nothing_NP (mkAdv except_Prep (mkNP a_Art plNum cn)) ;
  lt_Formula x y = {s = x.s ++ "<" ++ y.s} ;
  varTerm x = x ;

```

```

strVar s = s ;
termNP x = symb (ss x.s) ;
adjCN = mkCN ;
positAP = mkAP ;
relCN = mkCN ;
predRS rp vp = mkRS (mkRC1 rp vp) ;
slashRS rp np v2 = mkRS (mkRC1 rp np v2) ;
which_RP = Syntax.which_RP ;
vpS np vp = mkS (mkC1 np vp) ;
neg_vpS np vp = mkS negativePol (mkC1 np vp) ;
vVP = mkVP ;
vsVP = mkVP ;
v2VP = mkVP ;
apVP = mkVP ;
compVP a np = mkVP (mkAP a np) ;
canVP = mkVP can_VV ;
formulaS f = symb (ss f.s) ;
coordS = mkS ;
and_Conj = Syntax.and_Conj ;
or_Conj = Syntax.or_Conj ;
if_thenS = mkS if_then_Conj ;
possibleS s = mkS (adj_thatC1 possible_A s) ;
necessaryS s = mkS (adj_thatC1 necessary_A s) ;
}

```

A.3 The Core Attempto Interface

```

interface LexAttempto = open Syntax in {
oper
  possible_A : A ;
  necessary_A : A ;
  eachOf : NP -> NP ;
  adj_thatC1 : A -> S -> C1 ;
}

```

A.4 English, French, and German Implementations

```

instance LexAttemptoEng of LexAttempto =
  open SyntaxEng, ParadigmsEng, ExtraEng in {
oper
  possible_A = mkA "possible" ;
  necessary_A = mkA "necessary" ;
  eachOf np = mkNP (mkNP each_Det) (mkAdv part_Prep np) ;
  adj_thatC1 a s = mkC1 (mkVP (mkAP (mkAP a) s)) ;
}

instance LexAttemptoFre of LexAttempto = open ExtraFre,
  MakeStructuralFre, SyntaxFre, ParadigmsFre, Prelude in {
oper

```

```

possible_A = mkA "possible" ;
necessary_A = mkA "nécessaire" ;
eachOf np =
  mkNP (mkPredet "chacun" "chacune" genitive True) np ;
adj_thatCl a s =
  mkCl (mkVP (mkVP (mkAP a)) (SyntaxFre.mkAdv that_Subj s)) ;
}

instance LexAttemptoGer of LexAttempto = open ExtraGer,
  SyntaxGer, ParadigmsGer, MakeStructuralGer, Prelude in {
oper
  possible_A = mkA "möglich" ;
  necessary_A = mkA "nötig" ;
  eachOf np =
    mkNP (mkPredet (mkA "jed") "von" dative True singular) np ;
  adj_thatCl a s = mkCl (mkVP (mkAP (mkAP a) s)) ;
}

```

A.5 The Top-Level Core Grammar for English, French, and German

```

concrete AttemptoEng of Attempto = AttemptoI with
  (Syntax = SyntaxEng),
  (Symbolic = SymbolicEng),
  (LexAttempto = LexAttemptoEng) ;

concrete AttemptoFre of Attempto = AttemptoI with
  (Syntax = SyntaxFre),
  (Symbolic = SymbolicFre),
  (LexAttempto = LexAttemptoFre) ;

concrete AttemptoGer of Attempto = AttemptoI with
  (Syntax = SyntaxGer),
  (Symbolic = SymbolicGer),
  (LexAttempto = LexAttemptoGer) ;

```

A.6 The Test Lexicon

```

abstract TestAttempto = Attempto ** {
fun
  card_N : CN ;
  water_MCN : MCN ;
  rich_A : A ;
  customer_N : CN ;
  enter_V2 : V2 ;
  bank_N : CN ;
  important_A : A ;
  fill_in_V2 : V2 ;

```

```

    believe_VS : VS ;
    reject_V2 : V2 ;
    accept_V2 : V2 ;
}

```

```

concrete TestAttemptoEng of TestAttempto = AttemptoEng **
  open SyntaxEng, ParadigmsEng, IrregEng in {
lin
  card_N = mkkN "card" ;
  water_MCN = mkCN (mkkN "water") ;
  rich_A = mkA "rich" ;
  customer_N = mkkN "customer" ;
  enter_V2 = mkV2 "enter" ;
  bank_N = mkkN "bank" ;
  important_A = mkA "important" ;
  fill_in_V2 = mkV2 (partV (mkV "fill") "in") ;
  believe_VS = mkVS (mkV "believe") ;
  reject_V2 = mkV2 "reject" ;
  accept_V2 = mkV2 "accept" ;
oper
  mkkN : Str -> CN = \n -> mkCN (ParadigmsEng.mkkN n) ;
}

```

```

concrete TestAttemptoFre of TestAttempto = AttemptoFre **
  open SyntaxFre, ParadigmsFre, IrregFre in {
lin
  card_N = mkkN "carte" ;
  water_MCN = mkgN "eau" feminine ;
  rich_A = mkA "riche" ;
  customer_N = mkkN "client" ;
  enter_V2 = mkV2 "entrer" ;
  bank_N = mkkN "banque" ;
  important_A = mkA "important" ;
  expensive_A = mkA "cher" ;
  fill_in_V2 = mkV2 "remplir" ;
  believe_VS = mkVS (mkV croire_V2) ;
  reject_V2 = mkV2 "rejeter" ;
  accept_V2 = mkV2 "accepter" ;
oper
  mkkN : Str -> CN = \n -> mkCN (ParadigmsFre.mkkN n) ;
  mkgN : Str -> Gender -> CN = \n,g ->
    mkCN (ParadigmsFre.mkkN n g) ;
}

```

```

concrete TestAttemptoGer of TestAttempto = AttemptoGer **
  open SyntaxGer, ParadigmsGer, IrregGer in {

```

```
lin
  card_N = mkkN "Karte" ;
  water_MCN = mkgN "Wasser" "Wasser" neuter ;
  rich_A = mkA "reich" ;
  customer_N = mkgN "Kunde" "Kunden" masculine ;
  enter_V2 =
    mkV2 (mkV "ein" treten_V) (mkPrep "in" accusative) ;
  bank_N = mkgN "Bank" "Banken" feminine ;
  important_A = mkA "wichtig" ;
  fill_in_V2 = mkV2 (mkV "ab" (mkV "füllen")) ;
  believe_VS = mkVS (mkV "glauben") ;
  reject_V2 = mkV2 (fixprefixV "ver" werfen_V) ;
  accept_V2 = mkV2 (mkV "an" nehmen_V) ;
oper
  mkkN : Str -> CN = \n -> mkCN (ParadigmsGer.mkN n) ;
  mkgN : Str -> Str -> Gender -> CN = \s,n,g ->
    mkCN (ParadigmsGer.mkN s n g) ;
}
```

Polysemy in Controlled Natural Language Texts

Normunds Gruzitis and Guntis Barzdins

Institute of Mathematics and Computer Science, University of Latvia
normundsg@ailab.lv, guntis@latnet.lv

Abstract. Computational semantics and logic-based controlled natural languages (CNL) do not address systematically the word sense disambiguation problem of content words, i.e., they tend to interpret only some functional words that are crucial for construction of discourse representation structures. We show that micro-ontologies and multi-word units allow integration of the rich and polysemous multi-domain background knowledge into CNL thus providing interpretation for the content words. The proposed approach is demonstrated by extending the Attempto Controlled English (ACE) with polysemous and procedural constructs resulting in a more natural CNL named PAO covering narrative multi-domain texts.

1 Introduction

There are several sophisticated controlled natural languages (CNL), which cover relatively large subsets of English grammar, providing seemingly informal, high-level means for knowledge representation. CNLs typically support deterministic, bidirectional mapping to some formal language like first-order logic (FOL) or its decidable description logic (DL) subset [1], allowing integration with existing tools for reasoning, consistency checking or satisfiability model building.

Two widely accepted restrictions are used in CNLs to enable unambiguous construction of discourse representation structures: a set of interpretation rules for ambiguous syntactic constructions, and a monosemous lexicon — content words are not interpreted, they are treated as predicate identifiers whose meaning is defined only by FOL formulas derived from the text being analyzed¹. While the first restriction limits only syntactic sophistication of a language, the second one causes essential communication limitations, as the natural language lexicon is inherently polysemous.

The problem of CNLs like ACE [3] is that although they include a rich lexicon of content words, this lexicon is purely syntactic and has no meaning or interpretation within the CNL itself. Currently it is left up to the CNL user to define (in the CNL text) the difference between the content words like “take” and “give” — this vital background knowledge is not part of the CNL. But the problem is even deeper — even users cannot know in advance all the possible meanings that might be associated with a particular word like “take”: depending on the context “take” could mean

¹ This is true for ACE and alike CNLs. There are other, non-deterministic CNLs such as CPL [2], which perform shallow semantic analysis to interpret content words.

creating a photo with a camera, depriving a person from something, helping someone to get home, or something else. This is where the need for polysemy support in CNL arises — only through polysemy it is possible to incorporate multi-domain background knowledge into the CNL.

The paper is organized in two parts — the first part (Sections 2, 3, 4) introduces the vital concepts of micro-ontology and multi-word unit to systematically cope (in a controlled manner) with noun polysemy in the classic CNLs such as ACE (formally, OWL DL² subset of ACE). In the second part of the paper (Sections 5, 6, 7) we go further and define a new kind of expressive CNL (named PAO) combining the declarative (static) and procedural (dynamic) background knowledge expressed in OWL DL ontologies and SPARQL/UL procedural templates respectively — this will allow to provide adequate support also for verbs and their polysemy. The new PAO CNL is illustrated on a simple fairytale fragment with the basic temporal and spatial expressions.

By this paper we also want to raise the general awareness about the role of polysemy as a gateway for incorporating the rich background knowledge into CNLs. The root cause of polysemy is that there is a “finite” set of words in the language, but there is an “unlimited” set of concepts in various domains or contexts [4] that might need to be named in order to be referenced. Although new words are invented over the time as well, this happens comparatively rarely and slowly — the new words have to be accepted and learned by the community. Therefore reuse of existing well-known words in different contexts is a common “workaround”.

There are two main ways how words are reused [5]: metaphorically (relying on similarity, e.g. “mouse” for “computer pointing device”) and metonymically (relying on relationship, e.g. “library” for “building of library”). There are also homonyms — words that “accidentally” have the same spelling, but their meanings and origins are unrelated — they can be seen more like exceptions.

Thus metaphoric and metonymic reuse of existing lexemes is unavoidable in the natural language, what can be summed up in a saying: language is a graveyard of “dead” metaphors [6]. Fortunately, it is observed [7] that the various senses of the same lexeme typically fall into different domains — therefore explicit identification of these domains is what enables the controlled polysemy described in this paper.

2 OWL DL Compliant Micro-ontologies as a Sense Inventory

A monosemous lexicon (terminology) could be appropriate for descriptions that verbalize single-domain knowledge, i.e., consistent OWL DL ontologies. However, even seemingly consistent descriptions might run into lexical ambiguities. A possible solution in such cases is to introduce ad-hoc lexemes by explicitly pointing out the specific meanings (e.g. “library-building” versus “programming-library”), but dependency on such ad-hoc naming makes the language unsystematic and, thus, user-unfriendly [9]. Instead, we will demonstrate how multi-word units (MWU) can be created systematically and largely automatically while dynamically merging together terminology of different domains.

² In this paper we use term OWL DL, a subset of OWL, OWL 1.1 and OWL 2.0, to avoid reliance on the specific OWL version [8].

Internally monosemous and consistent domain ontologies that follow lexically motivated naming convention we will call *micro-ontologies*³. By introducing the concept of micro-ontology we provide a systematic solution to the lexical ambiguity problem — instead of trying to make all content words globally unique through ad-hoc lexemes, we suggest splitting the background knowledge into multiple, relatively small and lexically unambiguous micro-domains⁴ (see Fig.1). The benefit of this approach is its scalability to cover polysemous multi-domain terminology through a semi-automatic word sense partitioning procedure described in the next section.

By substituting a large, consistent ontology with numerous domain micro-ontologies we are inevitably introducing an ontology merging problem⁵. The problems of ontology merging (alignment, matching) and word sense disambiguation (WSD) are tightly intertwined and, in our view, the lack of definitive success in any of them is largely due to addressing these issues separately. Only by bringing both problems together it becomes possible to devise a method for semi-automatic *word sense partitioning* during micro-ontology merging process. The systematically partitioned word senses further enable semi-automatic *word sense disambiguation*.

	Micro-ontologies	
	Domain	Terminology (ontological text)
TBox	General	<i>Every building is a physical_entity.</i> <i>Every collection is an abstract_entity.</i> <i>No physical_entity is an abstract_entity.</i>
	Building	<i>Everything that has a roof is a building.</i> <i>Every library is a building.</i> <i>Every green_roof is a roof.</i>
	Programming	<i>Everything that contains something is a collection.</i> <i>Every library is a collection.</i> <i>Every function is something.</i>
ABox	Assertions (factual texts)	
	<i>There is a library_[building] that has a green_roof.</i>	
	<i>AbsoluteValue is a function. [...] The library_[programming] contains AbsoluteValue.</i>	

Fig. 1. The three micro-ontologies verbalized in ACE illustrate emergence of polysemy for the lexeme “library” appearing in the terminological and assertional statements. The appropriate sense (namespace) has to be explicitly assigned to each utterance of “library” to create a consistent merged ontology.

³ The proposed concept of micro-ontology bears some similarity to the concepts of environment or viewpoint in [10] and to the Cyc micro-theories [11], where all world-knowledge is split into narrow domain micro-theories. Meanwhile our word sense partitioning approach utilises standard OWL DL reasoning compatible with existing CNLs like ACE.

⁴ We intentionally avoid discussion about the optimal size of micro-ontologies, as this is formally irrelevant. The choice might be between larger domains like Education, Sports, Finance or much smaller domains like FrameNet [12] frames.

⁵ Although there is a vast literature on these issues (e.g. in [13]), majority of methods rely on shallow semantic similarity based on an external lexical taxonomy like WordNet [14]. Instead, we encourage to use the same OWL DL micro-ontologies as a sense inventory.

In the OWL DL subset of ACE-like CNLs, two kinds of statements can be differentiated [15]: terminological (ontological) and assertional (factual) ones, corresponding to the description logic TBox and ABox respectively. Ontological statements are those that introduce categories and describe relationships between them (“Every mouse is an animal”). Meanwhile factual statements are those talking about individuals belonging to specific categories (“The mouse_[computer] X is connected_to the workstation Y”).

In current CNLs both kinds of sentences usually are mixed into the same text: while populating facts one has to explicitly introduce also the ontology⁶. Instead, in our approach the two kinds of sentences must be strictly separated into *ontological texts* and *factual texts*. The rationale for separating these two kinds of texts is that ontological texts in our approach by definition are lexically monosemous within the scope of a single micro-ontology. Thus the WSD problem becomes limited only to the factual texts.

One has to remember that once a sufficient amount of background knowledge micro-ontologies are accumulated, the majority of the actual content will be lexically ambiguous multi-domain factual texts. By separating ontological and factual texts, our intention is to relieve an average CNL end-user from providing ontological statements (e.g. “Every library is a collection”), but rather allow the CNL end-user to concentrate on the factual content (“The library X contains a function Y”). The creation of domain micro-ontologies and their consistent merging (applying the semi-automatic word sense partitioning procedure described below) should be left to domain experts and knowledge engineers — as is the common practice already.

3 Word Sense Partitioning during Micro-ontology Merging

The formal semantics and decidability of OWL DL enables powerful means for what we call word sense partitioning. Namely, an OWL DL reasoner can automatically detect any formal inconsistencies (unsatisfiability) caused by incorrectly partitioned word senses during micro-ontology merging.

Note that in this section we consider only polysemy for class names (nouns) used within the micro-ontologies; property names are assumed to be globally monosemous among all micro-ontologies. This simplification will be justified in Section 5 where polysemy for verbs (predicates) will be introduced through procedural templates, making property names largely lexically invisible and, thus, less prone to problems of polysemy.

Assuming that all micro-ontologies reside in separate namespaces, a trivial approach to micro-ontology merging (possibly resulting in an inconsistent result) would be to add `owl:equivalentClass` axioms among all same-named classes from all micro-ontologies. This approach would have worked only if all lexemes used as class names were globally monosemous in all micro-ontologies.

In the presence of polysemous class names among different micro-ontologies, the actual merging procedure needs to avoid stating equivalence of same-named classes, if this would cause unsatisfiability of the merged ontology. Moreover, the problem is amplified by the possibility that the same name is used for classes, one of which

⁶ Of course, ontological sentences can be manually reused with other factual texts pertaining to the same domain.

actually is a subclass of the other (for example, “moon” in the astronomy micro-ontology might denote a satellite of any planet, while in the regular calendar micro-ontology it would more likely denote only the satellite of Earth). Such partitioning of same-named classes into separate meanings (senses) also requires introduction of unique, linguistically motivated sense identifiers to differentiate the distinct meanings of the same lexeme in the merged ontology; in the next section we will show how multi-word units can be used to address this issue systematically.

Assuming that in the given set of micro-ontologies there is a unique “conceptually correct” partitioning of senses for the polysemous class names, we would like to design a micro-ontology merging procedure that would find this partitioning automatically. Meanwhile one cannot have a free lunch: for the automatic procedure to work, the given set of micro-ontologies must satisfy the following strict conditions:

1. Micro-ontologies contain sufficient constraints to cause unsatisfiability, if relation `<X rdfs:subClassOf Y>` is inserted between any pair of same-named classes, which are unrelated in the “conceptually correct” partitioning of senses.
2. Micro-ontologies contain sufficient constraints to cause unsatisfiability, if for any pair of subsuming same-named classes `<X rdfs:subClassOf Y>` in the “conceptually correct” partitioning of senses an opposite relation `<Y rdfs:subClassOf X>` is inserted.

Under the above conditions there is an automatic merging procedure, which correctly partitions meanings of the polysemous class names and creates the merged ontology (sense inventory). The merging procedure relies on the fact that the axiom `<X owl:equivalentClass Y>` is equivalent to the conjunction of axioms `<X rdfs:subClassOf Y>` and `<Y rdfs:subClassOf X>`. At the start of the merging procedure all same-named classes are partitioned, because they belong to distinct namespaces of respective micro-ontologies. To create a merged ontology the merging procedure attempts inserting `rdfs:subClassOf` relation (in both directions) between all pairs of same-named classes across all micro-ontologies and checks satisfiability of the resulting ontology with an OWL DL reasoner after every such insertion — only those `rdfs:subClassOf` insertions that are not causing unsatisfiability are preserved. The final stage of this procedure is to insert `<X owl:equivalentClass Y>` for all pairs of same-named classes for which both `<X rdfs:subClassOf Y>` and `<Y rdfs:subClassOf X>` relations have been inserted in the previous stage.

The described micro-ontology merging (sense partitioning) procedure will produce a correct result only if the above conditions (1) and (2) are met. There is no automatic way to tell if the produced merging result is conceptually correct or wrong — checking the merging result is the duty of the knowledge engineer, who will generally need to laboriously debug and fine-tune the micro-ontologies until the conditions (1) and (2) are met and the automatic merging procedure produces a conceptually correct sense partitioning. Nevertheless this semi-automated method is superior to manual creation of the merged ontology, because it relies on local validity of every included micro-ontology and thus scales with adding new micro-ontologies by potentially many domain experts — a goal hardly achievable otherwise.

The described micro-ontology merging procedure with word sense partitioning has to be performed only when the set of micro-ontologies is updated (this set of micro-ontologies can be considered a background knowledge part of CNL). This means that running of this procedure (and potential debugging of micro-ontologies) is generally the task of domain experts and knowledge engineers designing and combining the micro-ontologies into the sense inventory, and not the task of the CNL end-user, typically involved only with semi-automatic disambiguation in the factual texts against the given sense inventory, as illustrated in the next section.

4 Example of Controlled Polysemy in ACE-OWL

To illustrate how the described micro-ontology approach would fit into an existing CNL like OWL subset of ACE (we will call it ACE-OWL for short), let us consider the following example factual sentence, which could have been entered by the ACE-OWL end-user:

A grandpa remembers [a] Germany that is not involved by [a] NATO.

The use of indefinite articles together with proper names in the above example is somewhat artificial and is done only in order to guide the ACE parsing engine⁷ to produce the below paraphrase where also proper names are treated as class names and not individuals. This is necessitated by the strict division of ontological background knowledge (introducing the terms, including proper names) and factual texts (using these terms for referring to specific individuals⁸). This approach is compensated by the uniform anaphora resolution detailed more in Section 7. Due to the strict separation of factual texts indefinite articles there could be assigned to proper names automatically and invisibly to the end-user.

There is a grandpa X1.

The grandpa X1 remembers a Germany X2.

It is false that a NATO involves the Germany X2.

To interpret the content of this factual text, a set of micro-ontologies describing the possibly polysemous meanings of appearing content words has to be defined in advance. Seven sample micro-ontologies are shown in Fig.2, describing some relevant background knowledge about geopolitics and people. The micro-ontologies themselves are presented in the regular ACE-OWL syntax [16]; the only “irregularity” is that the background knowledge is grouped into isolated, monosemous micro-ontologies, identified by the unique and explicitly stated namespaces. This allows to introduce polysemous meanings for content words such as “Germany” with the only restriction being that each meaning is described in a separate micro-ontology.

⁷ See <http://attempto.ifi.uzh.ch/site/tools/> (type all proper names in lower case with prefix n:).

⁸ This is obvious for common nouns, but is often true also for proper nouns, because “Germany”, for instance, could refer to “East Germany” or “West Germany”.

<i>Political map of Western Europe during the Cold War</i>	
(we:) http://m-ont.org/ColdWarWesternEurope.owl	
1.1	Every West_Germany is a country.
1.2	Every West_Germany is a Germany. Every Germany is a West_Germany.
1.3	Every NATO is an alliance.
1.4	Every West_Germany is involved by a NATO.

<i>Political map of Eastern Europe during the Cold War</i>	
(ee:) http://m-ont.org/ColdWarEasternEurope.owl	
2.1	Every Soviet_satellite_state is a country.
2.2	Every East_Germany is a Soviet_satellite_state.
2.3	Every East_Germany is a Germany. Every Germany is an East_Germany.
2.4	Every Warsaw_Pact is an alliance.
2.5	Every Soviet_satellite_state is involved by a Warsaw_Pact.

<i>Political map of Europe in 2007</i>	
(eu:) http://m-ont.org/Europe2007.owl	
3.1	Every federation is a country.
3.2	Every NATO_country is a country.
3.3	Every Germany is a federation.
3.4	Every Germany is a NATO_country.

<i>Bridging axioms for the political maps</i>	
(b1:) http://m-ont.org/b1.owl	
4.1	No {we,ee,eu,lg}:country that is involved by a we:NATO is involved by a ee:Warsaw_Pact.
4.2	Every eu:NATO_country is involved by a we:NATO.
4.3	No {we,ee,eu,lg}:country is an {we,ee,og}:alliance.
4.4	Every Prewar_Germany is a lg:Germany that is not involved by something that is a we:NATO or that is a ee:Warsaw_Pact.

<i>Map of official languages of European countries</i>	
(lg:) http://m-ont.org/language.owl	
5.1	Every German_speaking_country is a country.
5.2	Every Italian_speaking_country is a country.
5.3	Every Germany is a German_speaking_country.
5.4	Every Switzerland is a German_speaking_country.
5.5	Every Switzerland is an Italian_speaking_country.

<i>Taxonomy of organizational bodies</i>	
(og:) http://m-ont.org/organization.owl	
6.1	Every alliance is an organization.
6.2	Every alliance is a federation. Every federation is an alliance.

<i>Person name catalogue</i>	
(ps:) http://m-ont.org/person.owl	
7.1	Every man is a human.
7.2	Every woman is a human.
7.3	Every grandpa is a man.

Fig. 2. A set of sample micro-ontologies that describe small pieces of the changing political map of Europe. Few other micro-domains are sketched as well.

In cases where one micro-ontology needs to explicitly reference a class from another micro-ontology — a situation typical for “bridging” micro-ontologies (see *b1:* in Fig.2), a namespace identifier (or a list of identifiers) of the referenced remote

ontology(-ies) must be included. We shall remind (see Section 3) that polysemy here is limited to class names only — all properties appearing in micro-ontologies are assumed to be monosemous and globally shared.

Besides designing micro-ontologies, the task of the knowledge engineer is also to ensure consistent merging of selected micro-ontologies by applying the algorithm described in Section 3. In case of a semantically incorrect matching of polysemous terms, the knowledge engineer must complement the micro-ontologies (or create a bridging ontology) with additional constraints.

A merger of the micro-ontologies given in Fig.2 is shown in Fig.3 where the (inconsistent) meanings of ambiguous words “Germany” and “federation” are split by the word sense partitioning algorithm. Although more restrictions could be added to separate today’s “Germany” and “West_Germany” into different meanings, the current merger is sufficient for our demonstration.

Fig.3 also illustrates the use of multi-word units (MWU) to identify senses of the polysemous lexemes. MWU is a common technique used in natural language to differentiate meanings of polysemous lexemes (e.g. “computer mouse” to specify a meaning of the polysemous lexeme “mouse”). We use dashes to explicitly identify the two parts forming MWUs (e.g. “computer-mouse”). In our case the second part of a MWU is the original lexeme (including compound lexemes). As the first part of the MWU we recommend to use the name (namespace prefix) of one of the micro-ontologies, where this particular meaning appears. It should be noted that a MWU have to be created only in case of a polysemous lexeme.

The benefit of this approach is that self-explanatory MWUs can be created automatically, provided that micro-ontology names are linguistically motivated. Although it is questionable whether self-explanatory names can be provided for all micro-ontologies, this should be achievable in relatively isolated worlds. An alternative option is that the knowledge engineer manually defines an equivalent class with a more specialised name (term) which may be used for identification of the particular word sense.

Assuming that these tasks of the knowledge engineer are completed and the merged ontology with partitioned meanings of the polysemous lexemes is obtained, we can proceed with the parsing of the ACE-OWL factual text shown in the beginning of this section. The ACE parsing engine is used for the initial parsing of the input text, followed by a separate WSD step, during which the polysemous lexemes (in our example — “Germany”) must be matched against the same-named classes (ignoring prefixes) of the merged ontology (Fig.3). Semi-automatic detection of a valid sense match (or possibly several valid sense matches) again is enabled by an OWL reasoner — a sense match is considered valid, if the resulting ontology (merged micro-ontologies and OWL statements corresponding to the factual input text) is still satisfiable. In case of the provided input text, only the following reading thus is found valid:

A grandpa remembers [a] ColdWarEasternEurope–Germany that is not involved by [a] NATO.

In general, the described WSD procedure might work only partially (leaving multiple options) due to insufficiently rich context provided in the factual input text. Therefore there should be an alternative for the CNL end-user to choose the

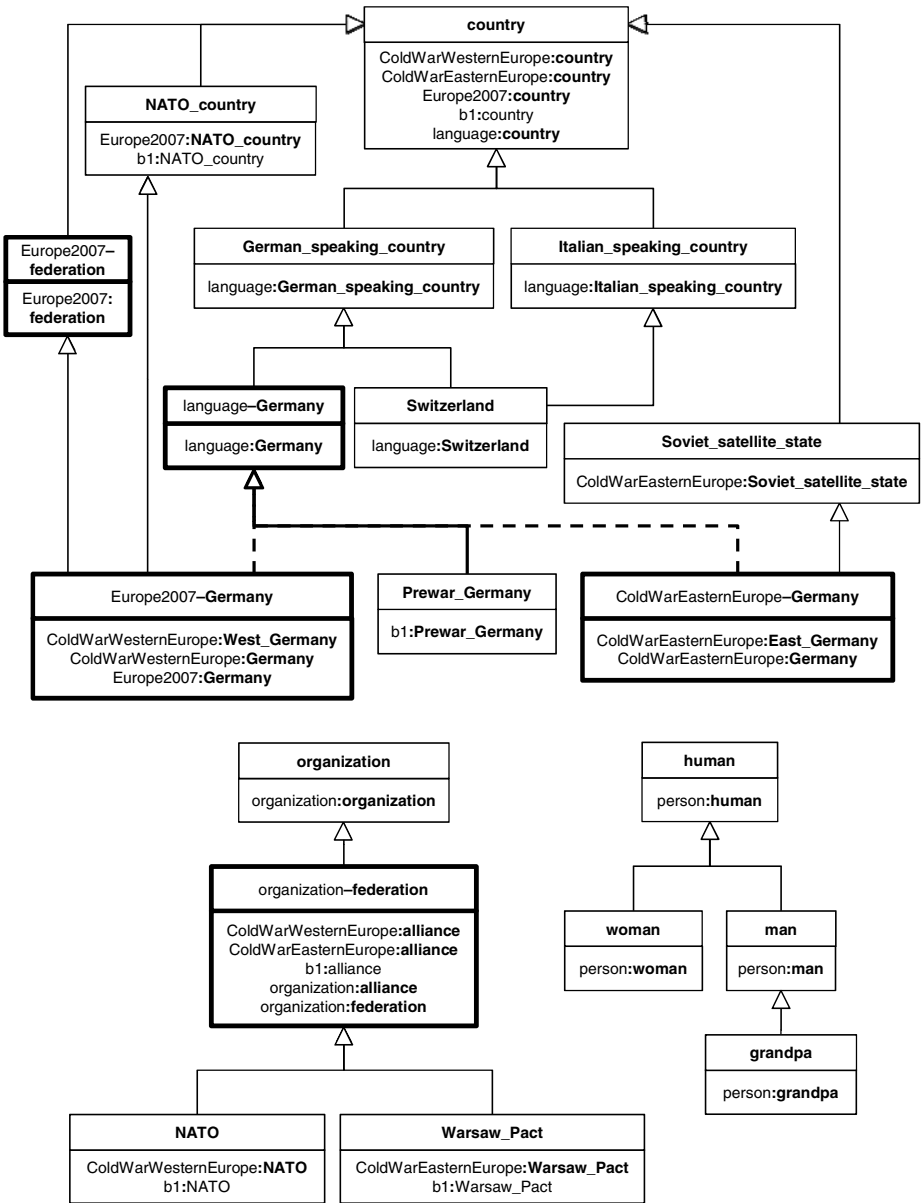


Fig. 3. Merged micro-ontologies from Fig.2 with word senses partitioned as per algorithm described in Section 3. Each vertex in the graph represents a merged class: the lower section lists qualified names of the matched classes; the upper section shows name to be used for the merged class, which in case of polysemous lexeme may be a MWU. The dashed edges illustrate automatically detected subsumption relationships between senses of the same lexeme.

appropriate paraphrase manually — the automatic “guessing” technique would more likely be used only as a hinting engine for the CNL end-user, helping to select the meaning of a polysemous lexeme.

In contrast to the legacy macro-ontologies, micro-ontologies offer several significant advantages: (i) they do not impose a single consistent scheme, allowing many distinct points of view to co-exist; (ii) they can be seen as snapshots of some aspects of “reality”, supporting non-stable and temporal entities; and (iii) they scale well — reality do not have to be compressed into a restricted number of lexical categories thus avoiding “signal losses” during conceptualisation.

5 PAO — A CNL with Polysemous and Procedural Constructs

In the previous part of the paper we tried to stay conservative and apply only minimum changes to the traditional ACE-OWL while adding polysemy for class-names and background knowledge support. In the following part of the paper we will try to stretch the limits and go beyond the traditional ACE capabilities in order to provide an adequate polysemy support also for verbs. We will do so by defining a new CNL named PAO (for Procedural ACE-OWL).

Besides polysemy support, the key difference of PAO is that it adds support also for procedural background knowledge in addition to the traditional declarative OWL ontology background knowledge. The distinction between procedural actions and declarative properties is neglected in OWL and related CNLs. Although temporal OWL ontologies are sometimes used, one has to remember that OWL as a subset of FOL has no native time concept and can introduce time only as part of the model; one has to go outside FOL and use procedural means to define model-changing-actions as will be illustrated with the execution trace in Section 7. In order to stay within the familiar OWL and RDF realm, SPARQL/UL⁹ is an obvious choice for the procedural background knowledge component. In PAO we stay short of the full-fledged procedural language with IF-THEN-ELSE, GOTO and similar control features — these could be added, but are not necessary for the simple narrative text describing a plain sequence of actions such as a simple fairytale fragment used below to introduce the core PAO principles.

An additional rationale for combining OWL and SPARQL/UL is that only together they provide the expressive power equivalent to the regular relational database programming — ER-diagrams of relational database schemas (OWL equivalent) would be of little use without SQL language (SPARQL/UL equivalent). As we will see, this expressive power is adequate also for supporting narrative texts in a CNL.

PAO defines a natural and unambiguous translation from the controlled language input text into combination of OWL and SPARQL/UL. Formal definition of PAO is beyond the scope of this paper, but we will describe PAO by a detailed example covering its core features. Despite a rather different approach taken to define PAO, classic ACE-OWL technically will remain a subset of PAO.

⁹ We will use the abbreviation SPARQL/UL for a union of SPARQL and SPARQL/Update — RDF query and update languages, now both part of SPARQL 1.1 [17].

6 Defining the Background Knowledge in PAO

The background knowledge necessary for interpretation of a CNL text can be described in the controlled language itself, or it can be imported from some other formalism. In PAO we permit both, but encourage the second approach due to potentially large size and complex structure of the background knowledge and, since also in real-life, background knowledge is often learned non-verbally through diagrams, examples or other means.

In PAO background knowledge consists of two parts — declarative OWL micro-ontologies (Fig.4) and procedural templates (Fig.6). The purpose of micro-ontologies (similarly to example in Section 4) is to define the *concept hierarchies* (OWL classes), their *relationships* (OWL properties) and *restriction axioms* (cardinality restrictions and others). The main design principle of micro-ontologies is to keep them lexically oriented to enable direct mapping of class (and optionally also property names) to the content words in the CNL text. Note that background knowledge is a pure TBox and never includes any OWL individuals (creation of ABox individuals will happen during anaphora resolution process) — for this reason in People micro-ontology there is introduced a subclass LittleRedRidingHood for all people (possibly just one) having this name — this approach enables uniform anaphora resolution mechanism to be used later. In PAO it is also required that each micro-ontology has a unique name (namespace). All class-names used in the micro-ontologies in Fig.4 have unique meaning (including the two meanings of lexeme “basket”) and, therefore, the automatic micro-ontology merging procedure from Section 3 here results in a simple union of the shown background knowledge input ontologies.

In Fig.4 micro-ontologies are visualized according to the UML profile for OWL standard [18]. Alternatively, micro-ontologies may be defined verbally through ACE-OWL as illustrated in Fig.5 — this allows ACE-OWL ontological sentences (sentences dealing only with TBox) to remain a subset of PAO. The part of ACE-OWL dealing with ABox will be discussed in the next section.

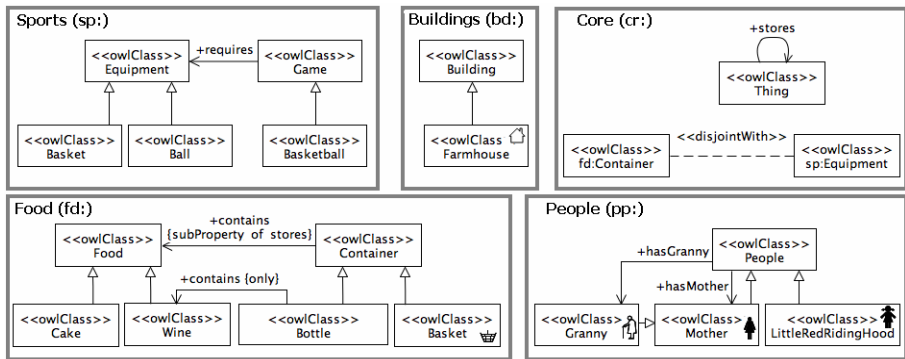


Fig. 4. Background knowledge micro-ontologies in UML profile for OWL syntax

Every Basket is a Container.
Every Bottle is a Container.
Every Cake is a Food.
Every Wine is a Food.
Everything that contains something is a Container.
Everything that is contained by something is a Food.
Everything that is contained by a Bottle is a Wine.
If X contains Y then X stores Y.

Fig. 5. Example of ACE-OWL verbalization of Food micro-ontology from Fig. 4

The purpose of procedural templates background knowledge in Fig.6 is to provide a link between the action words (lexical units representing verbs) and their “meaning” in SPARQL/UL. As mentioned, the distinction between actions and properties is often neglected, but in PAO they are strictly separated: in PAO action is a non-ontological SPARQL/UL procedure, which *creates/deletes* individuals or *connects/disconnects* them through the predefined properties. PAO action, unlike binary OWL properties, has no arity restriction — it can link any number of arguments as is typical for verbs in natural language.

Procedure: Residence

```

:parameters (?resident ?co-resident ?location)
:precondition ()
:effect (and(stores ?location ?resident)
        (stores ?location ?co_resident))
:lexicalUnits (camp, inhabit, live, lodge, reside, stay)

```

Procedure: Removing

```

:parameters (?agent ?source ?theme)
:precondition (stores ?source ?theme)
:effect (and(stores ?agent ?theme)
        (not(stores ?source ?theme)))
:lexicalUnits (confiscate, remove, snatch, take, withdraw)

```

Procedure: Bringing

```

:parameters (?agent ?goal ?theme)
:precondition (and(stores ?agent ?theme)
        (stores ?a ?agent) (not(= ?a ?goal)))
:effect (and(stores ?goal ?theme)(stores ?goal ?agent)
        (not(stores ?agent ?theme))
        (not(stores ?a ?agent)))
:lexicalUnits (bring, carry, convey, drive, haul, take)

```

Fig. 6. Procedural templates of background knowledge

Syntactically a procedural template in PAO is a combination of elements inspired by FrameNet [12], Planning Domain Description Language (PDDL) [19] for situation calculus, and SPARQL/UL. The procedural template itself corresponds to FrameNet frame, the parameters section corresponds to FrameNet frame elements (only the actually used elements are included), and the lexical units section is a direct copy from FrameNet. Inclusion of precondition and effect sections in the procedural

template is inspired by PDDL and has two-fold purpose: this is a compact representation of SELECT, INSERT, DELETE, MODIFY and WHERE patterns of the corresponding SPARQL/UL statement and at the same time it preserves compatibility with PDDL for planning purposes. Elements of planning will become necessary in the final steps of PAO interpretation described later. For word sense disambiguation purposes each procedural template must have a unique name.

Theoretically the precondition and effect sections of procedural templates could reference any class or property within any micro-ontology, but to achieve their maximum reusability, in PAO procedural templates are recommended to directly reference only universal properties with unrestricted domain and range — in Fig.4 they are depicted in a separate micro-ontology named “Core”, which contains also the necessary bridging axioms between the micro-ontologies. If a non-universal property needs to be manipulated by a procedural template, it can be defined as subproperty of some universal property (like “contains” is defined as a subproperty of the universal property “stores” in Fig.4).

The micro-ontologies and procedural templates shown in Fig.4 and Fig.6 are specifically crafted for the PAO example in the next section; for more realistic applications it would be necessary to create a much larger collections of micro-ontologies and procedural templates covering the whole lexicon and domain-knowledge of interest.

7 Example of PAO Text Processing

In this Section we are considering only narrative factual texts after the background knowledge (possibly including some ACE-OWL ontological text) has already been added into the system. Narrative texts in PAO have to be written in simple present tense to avoid complex event sequencing — the described events are assumed to be atomic and to occur sequentially as they are mentioned in the text. The following input text will be used in this paper to illustrate all processing stages of PAO.

LittleRedRidingHood lives in a farmhouse with her mother. She takes a basket from the farmhouse and carries it to her granny.

This input text is ambiguous in at least three ways assuming that the background knowledge is limited to that defined in Fig.4 and Fig.6 in the previous section:

1. Anaphora “she” could refer to Little Red Riding Hood or to her mother,
2. “basket” could refer to the *food-basket* or *sports-basket*,
3. “take” could refer to *removing* or *bringing* procedure template.

In the first analysis stage PAO assists the user with rephrasing such ambiguous text into unambiguous paraphrase shown in Fig.7. PAO paraphraser automatically identifies ambiguities, generates the possible multi-word units for them (by prepending a micro-ontology or a procedural template name from the available background knowledge) and asks the user to select the correct alternative, which is then recorded into the paraphrase. Similarly PAO paraphraser asks the user to select the correct antecedent for unclear anaphors.

- A. *Obj4 is a LittleRedRidingHood.*
- B. *Obj4 **lives** in obj8 with obj11.*
- C. *Obj8 is a farmhouse.*
- D. *Obj4 hasMother obj11.*
- E. *Obj4 **removing-takes** obj15 from obj8.*
- F. *Obj15 is a **food-basket**.*
- G. *Obj4 carries obj15 to obj25.*
- H. *Obj4 hasGranny obj25.*

Fig. 7. A disambiguated paraphrase of the input text

Technically the PAO paraphraser is a syntactic parser complemented with matching rules towards a fixed list of atomic paraphrase patterns — this step is largely similar to ACE-OWL paraphrasing illustrated in Section 4 and to techniques used by automated FrameNet annotators [20, 21]. The selected multi-word units and anaphora antecedents can be inserted also in the original text to make it unambiguous, e.g. “She-LittleRedRidingHood removing-takes a food-basket from the...”. Multi-word units are inserted only for the polysemous words.

Note that in the generated paraphrase in Fig.7 the statements A, C, D, F, and H are actually regular ACE-OWL factual statements about individuals within the background knowledge micro-ontologies. As such, these statements can be translated into corresponding OWL statements (actually RDF triples) by regular ACE-OWL means and these RDF triples then can be directly added into RDF database with a simple SPARQL/UL INSERT statement as shown in Fig.8. This procedure also ensures that PAO includes ACE-OWL factual statements (ABox) as a subset.

Meanwhile the procedural statements B, E, and G do not belong to ACE-OWL and require a procedural template from the background knowledge for their translation. The translation includes mapping of syntactic roles into procedural template parameters — such mapping techniques have been developed for automatic FrameNet annotation [20, 21] and their deterministic variation could be adapted also here. Next, the precondition and effect notation in the procedural template is translated into equivalent SPARQL/UL statement. In this second analysis stage paraphrase of the original input text gets converted into the sequence of SPARQL/UL statements shown in the first column of Fig.8.

Fig.8 contains also a second column which is filled in the third analysis stage. It contains SPARQL/UL statements, which implicitly follow from contents of the first column and the background knowledge in Fig.4 and Fig.6 — these statements are generated by the system automatically through OWL entailment and through PDDL-like planning over the preconditions and effects of the invoked procedural templates. The planning stage is needed here because in the input text some obvious intermediate steps of action might often be omitted and they need to be filled-in by planning to satisfy the procedural template preconditions — in our example for Little Red Riding Hood to be able to take a basket from the farmhouse, the basket had to be at the farmhouse in the first place. Here we leave entailment and planning explanation at the example level, but a more formal PAO definition would have to strictly limit the permitted extent of planning and entailment (for example, whether to allow here full

	EXPLICIT STATEMENTS	IMPLICIT STATEMENTS BY ENTAILMENT AND PLANNING
A	INSERT {<obj4> <rdf:type> <pp:LittleRedRidingHood>}	
B	INSERT {<obj8> <stores> <obj4>. <obj8> <stores> <obj11>}	
C	INSERT {<obj8> <rdf:type> <bd:Farmhouse>}	INSERT {<obj8> <stores> <obj15>} <i>Inserted by planning because of procedural template precondition at step E.</i>
D	INSERT {<obj4> <pp:hasMother> <obj11>}	INSERT {<obj11> <rdf:type> <pp:Mother>} <i>Entailed by range of the property pp:hasMother.</i>
E	DELETE {<obj8> <stores> obj15} INSERT {<obj4> <stores> <obj15>}	
F	INSERT {<obj15> <rdf:type> <fd:Basket>}	
G	DELETE {<obj4> <stores> <obj15>. ?a <stores> <obj4>} INSERT {<obj25> <stores> <obj15>. <obj25> <stores> <obj4>} WHERE {?a <stores> <obj4>. FILTER (?a != <obj25>)}	
H	INSERT {<obj4> <pp:hasGranny> <obj25>}	INSERT {<obj25> <rdf:type> <pp:Granny>} <i>Entailed by range of the property pp:hasGranny.</i>

Fig. 8. Generated SPARQL/UL statements (procedural statements are in bold)

OWL 2.0 reasoning over background knowledge micro-ontologies or to suffice with simpler RDFS entailment).

The fourth analysis stage is to execute the SPARQL/UL statement sequence shown in Fig.8 and to generate the RDF database content trace of such execution — Fig.9 shows the resulting stepwise RDF database content trace. Such content trace could technically be stored as a sequence of RDF named graphs, along with an additional named graph storing the background knowledge micro-ontologies.

The generated RDF database content trace is the final result of PAO text analysis — this trace is the actual discourse conveyed by the PAO input text. In the right column of Fig.9 the discourse is optionally visualized also as a sequence of graphic scenes — similarly to text-to-scene animation approach described in [22]. These visualizations can be generated automatically from the graphic icons provided for OWL classes in the background knowledge (Fig.4 includes the necessary icons); relations are visualized as labelled arrows or alternatively spatial relations like “stores” can be visualized as graphic inclusion. These visual scenes highlight the similarity of PAO analysis result to the dynamic scene likely imagined by a human reader incrementally reading the same input text.



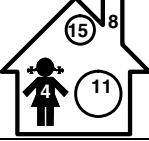



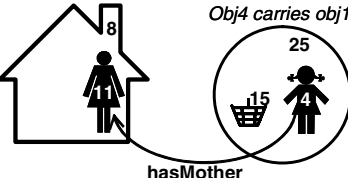
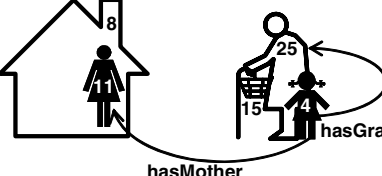
A	<code><obj4> <type> <LittleRedRidingHood>.</code> 	 <i>Obj4 is a LittleRedRidingHood.</i>
B	<code><obj4> <type> <LittleRedRidingHood>.</code> <code><obj8> <stores> <obj4>.</code> <code><obj8> <stores> <obj11>.</code>	 <i>Obj4 lives in obj8 with obj11.</i>
C	<code><obj4> <type> <LittleRedRidingHood>.</code> <code><obj8> <stores> <obj4>.</code> <code><obj8> <stores> <obj11>.</code> <code><obj8> <type> <farmhouse>.</code> <code><obj8> <stores> <obj15></code>	 <i>Obj8 is a farmhouse.</i>
D	<code><obj4> <type> <LittleRedRidingHood>.</code> <code><obj8> <stores> <obj4>.</code> <code><obj8> <stores> <obj11>.</code> <code><obj8> <type> <farmhouse>.</code> <code><obj4> <hasMother> <obj11>.</code> <code><obj11> <type> <mother>.</code> <code><obj8> <stores> <obj15></code>	 <i>Obj4 hasMother Obj11.</i> hasMother
E	<code><obj4> <type> <LittleRedRidingHood>.</code> <code><obj8> <stores> <obj4>.</code> <code><obj8> <stores> <obj11>.</code> <code><obj8> <type> <farmhouse>.</code> <code><obj4> <hasMother> <obj11>.</code> <code><obj11> <type> <mother>.</code> <code><obj4> <stores> <obj15></code>	 <i>Obj4 removing-takes obj15 from obj8.</i> hasMother
F	<code><obj4> <type> <LittleRedRidingHood>.</code> <code><obj8> <stores> <obj4>.</code> <code><obj8> <stores> <obj11>.</code> <code><obj8> <type> <farmhouse>.</code> <code><obj4> <hasMother> <obj11>.</code> <code><obj11> <type> <mother>.</code> <code><obj4> <stores> <obj15></code> <code><obj15> <type> <food-basket></code>	 <i>Obj15 is a food-basket.</i> hasMother
G	<code><obj4> <type> <LittleRedRidingHood>.</code> <code><obj25> <stores> <obj4>.</code> <code><obj8> <stores> <obj11>.</code> <code><obj8> <type> <farmhouse>.</code> <code><obj4> <hasMother> <obj11>.</code> <code><obj11> <type> <mother>.</code> <code><obj25> <stores> <obj15>.</code> <code><obj15> <type> <food-basket>.</code>	 <i>Obj4 carries obj15 to obj25.</i> hasMother
H	<code><obj4> <type> <LittleRedRidingHood>.</code> <code><obj25> <stores> <obj4>.</code> <code><obj8> <stores> <obj11>.</code> <code><obj8> <type> <farmhouse>.</code> <code><obj4> <hasMother> <obj11>.</code> <code><obj11> <type> <mother>.</code> <code><obj25> <stores> <obj15>.</code> <code><obj15> <type> <food-basket>.</code> <code><obj4> <hasGranny> <obj25>.</code> <code><obj25> <type> <granny></code>	 <i>Obj4 hasGranny Obj25.</i> hasMother

Fig. 9. RDF database content trace and its optional visualization

The constructed RDF database trace in Fig.9 can further be used to answer queries about the input text, for example:

1. *Who delivered a basket to a granny?*
2. *Did LittleRedRidingHood visit her granny?*
3. *Where initially was the basket?*
4. *When did the granny got the basket?*

To see how these queries could be answered through the constructed RDF database content trace, they first need to be disambiguated and with the same PAO techniques translated into the following SPARQL queries extended with the non-standard trace step identification in the WHERE-AT-STEP section (technically this non-standard trace step identification could be implemented through more lengthy RDF named graphs manipulation):

1.

```
SELECT ?x
WHERE-AT-STEP(?n) {?w <stores> ?x. ?x <stores> ?y.}
WHERE-AT-STEP(?n+1) {
  ?z <stores> ?x. ?z <stores> ?y.
  ?y <rdf:type> <fd:Basket>.
  ?z <rdf:type> <pp:Granny>}
```
2.

```
SELECT * WHERE-AT-STEP(any) {
  ?z <stores> ?x.
  ?x <rdf:type> <pp:LittleRedRidingHood>.
  ?z <rdf:type> <pp:Granny>}
```
3.

```
SELECT ?x WHERE-AT-STEP(min) {
  ?x <stores> ?y.
  ?y <rdf:type> <fd:Basket>}
```
4.

```
SELECT ?n WHERE-AT-STEP(?n) {
  ?y <stores> ?x.
  ?x <rdf:type> <fd:Basket>.
  ?y <rdf:type> <pp:Granny> }
```

The answers produced by these queries on the RDF trace in Fig.9 would be:

1. ?x = obj4
2. yes
3. ?x = obj8
4. ?n = H

These very technical SPARQL answers could afterwards be rendered into more verbose answers:

1. *LittleRedRidingHood [delivered a basket to granny].*
2. *Yes [, LittleRedRidingHood visited granny].*
3. *[Basket initially was] in the farmhouse.*
4. *In step H [, when LittleRedRidingHood brought the basket to granny].*

Although we have not described the question answering process here in detail, these examples provide an overview of PAO potential for factual and temporal question answering over narrative input texts.

8 Conclusion

We have shown that noun polysemy can be introduced into CNLs in a controlled manner through dividing the underlying terminological ontology into monosemous micro-ontologies. This not only provides a consistent naming for different senses of the same word through prefixing it with the source micro-ontology name, but also allows automatic OWL DL reasoning techniques to be employed for identifying and merging the non-conflicting uses of the same word in different micro-ontologies. The key benefit of this approach is that terminology of a particular domain is localised to the corresponding micro-ontology and the same words in this way can be freely reused in other micro-ontologies covering other parts of the background knowledge. In this way it becomes possible to bundle the rich and extensible background knowledge about content words within the CNLs like ACE.

Meanwhile to properly cover also verb (action, property) polysemy, we had to introduce a new CNL named PAO. The described PAO controlled language is only a rather simple attempt to exploit the rich declarative and procedural background knowledge in controlled natural language. We are quite pleased to have managed to include ACE-OWL as a proper subset of PAO thus achieving a complete complementary integration of procedural and declarative approaches. Also the briefly mentioned optional visualization of PAO discourse is a tempting area for further exploration — inversion of the mentioned visualization technique could lead to a vision simulation grounded in the same ontological and procedural background knowledge.

An obvious limitation of the presented PAO language is its treatment of time only as a linear sequence of events mentioned in the input text. A richer time conceptualization is generally needed, including hypothetical, parallel and negated events [23] to handle texts like “Mother told LittleRedRidingHood to go directly to the granny’s house and not to engage in conversations with strangers”.

Nevertheless, the described approach to polysemy in our view provides a new insight into how background knowledge can be systematically added into CNLs and how multiple kinds of background knowledge (procedural and declarative) extend CNL expressivity.

Acknowledgements. The underlying project is funded by the National Research Program in Information Technologies (Project No. 2). We also thank the reviewers of the CNL 2009 workshop proceedings for their valuable comments.

References

1. Schwitter, R., Kaljurand, K., Cregan, A., Dolbear, C., Hart, G.: A Comparison of three Controlled Natural Languages for OWL 1.1. In: 4th International OWLED Workshop (2008)
2. Clark, P., Harrison, P., Jenkins, T., Thompson, J., Wojcik, R.H.: Acquiring and Using World Knowledge Using a Restricted Subset of English. In: 18th International FLAIRS Conference, pp. 506–511 (2005)
3. Fuchs, N.E., Kaljurand, K., Schneider, G.: Attempto Controlled English Meets the Challenges of Knowledge Representation, Reasoning, Interoperability and User Interfaces. In: 19th International FLAIRS Conference (2006)

4. Pustejovsky, J.: Type Construction and the Logic of Concepts. In: Bouillon, P., Busa, F. (eds.) *The Language of Word Meaning*. Cambridge University Press, Cambridge (2001)
5. Rabin, Y., Leacock, C.: *Polysemy*. Oxford University Press, Oxford (2000)
6. Leary, D.E. (ed.): *Metaphors in the history of psychology*. Cambridge University Press, Cambridge (1994)
7. Magnini, B., Strapparava, C., Pezzulo, G., Gliozzo, A.: The role of domain information in word sense disambiguation. *Natural Language Engineering* 8(4), 359–373 (2002)
8. Web Ontology Language (OWL). W3C Recommendation (2009), <http://www.w3.org/TR/owl2-overview/>
9. Rinaldi, F., Dowdall, J., Hess, M., Molla, D., Schwitter, R., Kaljurand, K.: Knowledge-Based Question Answering. In: Palade, V., Howlett, R.J., Jain, L. (eds.) *KES 2003. LNCS (LNAI)*, vol. 2773, pp. 785–792. Springer, Heidelberg (2003)
10. Wilks, Y., Barnden, J., Wang, J.: Your metaphor or mine: belief ascription and metaphor interpretation. In: *12th International Joint Conference on Artificial Intelligence*, pp. 945–950 (1991)
11. Lenat, D.: Cyc: A Large-Scale Investment in Knowledge Infrastructure. *Communications of the ACM* 38(11), 33–38 (1995)
12. Fillmore, C.J., Johnson, C.R., Petruck, M.R.L.: Background to FrameNet. *International Journal of Lexicography* 16, 235–250 (2003)
13. Euzenat, J., Shvaiko, P.: *Ontology Matching*. Springer, New York (2007)
14. Banek, M., Vrdoljak, B., Tjoa, A.M.: Word Sense Disambiguation as the Primary Step of Ontology Integration. In: Bhowmick, S.S., Küng, J., Wagner, R. (eds.) *DEXA 2008. LNCS*, vol. 5181, pp. 65–72. Springer, Heidelberg (2008)
15. Schwitter, R.: Creating and Querying Linguistically Motivated Ontologies. In: *Knowledge Representation Ontology Workshop, Conference in Research and Practice in Information Technology*, vol. 90, pp. 71–80 (2008)
16. Kaljurand, K., Fuchs, N.E.: Verbalizing OWL in Attempto Controlled English. In: *3rd International OWLED Workshop* (2007)
17. SPARQL 1.1 Query and Update Language for RDF. W3C Working Draft (2009), http://www.w3.org/2009/sparql/wiki/Main_Page
18. Ontology Definition Metamodel. OMG Adopted Specification (2009), <http://www.omg.org/docs/ptc/07-09-09.pdf>
19. PDDL - The Planning Domain Definition Language. Technical report, Yale Center for Computational Vision and Control (1998), <http://www.cs.yale.edu/homes/dvm/>
20. Erk, K., Pado, S.: Shalmaneser - a flexible toolbox for semantic role assignment. In: *5th International LREC Conference* (2006)
21. Johansson, R., Nugues, P.: Comparing dependency and constituent syntax for frame-semantic analysis. In: *6th International LREC Conference* (2008)
22. Johansson, R., Berglund, A., Danielsson, M., Nugues, P.: Automatic text-to-scene conversion in the traffic accident domain. In: *19th International Joint Conference on Artificial Intelligence*, pp. 1073–1078 (2005)
23. Schubert, L.K., Hwang, C.H.: Episodic Logic meets Little Red Riding Hood: A comprehensive, natural representation for language understanding. In: Iwanska, L., Shapiro, S.C. (eds.) *Natural Language Processing and Knowledge Representation: Language for Knowledge and Knowledge for Language*, pp. 111–174. MIT/AAAI Press, Cambridge/Menlo Park (2000)

Economical Discourse Representation Theory

Johan Bos

`bos@meaningfactory.com`

1 Introduction

First-order logic (FOL) is undecidable — that is, no algorithm exists that can decide whether a formula of FOL is valid or not. However, there are various fragments of FOL that are known to be decidable. FO^2 , the two-variable fragment of FOL, is one of such languages [1,2]. FO^2 is a first-order language where formulas have maximally two variables, no function symbols, but possibly do have equality. FO^2 has the finite model property [1], which means that if a formula of FO^2 is satisfiable, it is satisfiable in a finite model.

In this paper we propose a controlled fragment of Discourse Representation Theory (DRT, [3]) with a semantics formalised on the basis of the two-variable fragment with equality. DRT encapsulates the idea of text interpretation that “one and the same structure serves simultaneously as content and context” [4], where *content* refers to the semantic interpretation of sentences already processed, and *context* serves in aiding the interpretation of anaphoric expressions (such as pronouns) in subsequent sentences.

However, providing a two-variable natural language fragment is, in itself, not a new idea. In fact, the framework presented here is very much inspired by Ian Pratt-Hartmann’s language E2V [5]. But as Pratt-Hartmann himself notes, E2V is “certainly not proposed as a practically useful controlled language”. Our aim is to try to find out how useful a controlled language based on the two-variable fragment actually can be, mostly from a computational linguistic point of view. We will do this by:

- defining a transparent translation from the DRT fragment to FO^2 ;
- including events, thematic roles, and pronouns in the fragment;
- specifying a syntax-semantics interface.

The syntax-semantic interface of our choice will be based on combinatory categorial grammar (CCG, [6]), rather than, say, a simple definite clause grammar. Because of its type transparency principle, CCG will enable us to set up a clean interface, where each syntactic category uniformly corresponds to a semantic type. CCG gives us further means for incremental processing [6], and large databases of texts annotated with CCG-derivations are available [7], which might aid us in enlarging the lexicon for practical applications.

This article is organised as follows. First we briefly revise Discourse Representation Theory (Section 2). Then we introduce our decidable fragment of DRT, Economical DRT (Section 3), including the syntax and semantics of the

meaning representation language. In Section 4 we show how a neo-Davidsonian analysis of events combines well with the main ideas of Economical DRT. We show how a syntax-semantics interface based on CCG can be realised for EDRT in Section 5. Finally, in Section 6, we discuss the interpretation of pronouns in multi-sentential texts.

2 Discourse Representation Theory (DRT)

DRT is a theory of natural language meaning, and was originally designed by Kamp to study the relationship between indefinite noun phrases and anaphoric pronouns as well as temporal relations [8]. Since its publication in the early 1980s, DRT has grown in depth and coverage, establishing itself as a well-documented formal theory of meaning, dealing with a large number of semantic phenomena ranging from pronouns, abstract anaphora, presupposition, tense and aspect, propositional attitudes, ellipsis, to plurals [3,9,10,11,4,12,13].

A key idea of DRT is that a DRS (Discourse Representation Structure) plays both the role of content (giving a precise model-theoretic interpretation of the text processed so far) and context (assisting in interpreting anaphoric expression occurring in subsequent text). Viewed from a distance, DRT can be seen as a theory with three components:

1. A formal language of DRS, the meaning representations for texts;
2. the syntax-semantics interface, mapping text into DRS;
3. A component that deals with the semantic interpretation of DRSs.

A DRS consists of a set of discourse referents and a set of DRS-conditions. DRS-conditions can either be basic, storing information about the discourse referents or relations between them, or complex, containing embedded DRSs. Hence DRSs are recursively defined, and the way they are nested predicts which discourse referents are accessible for future anaphoric reference and which are not. An example DRS is shown in Fig. 1, presented in the familiar box notation known from DRT.

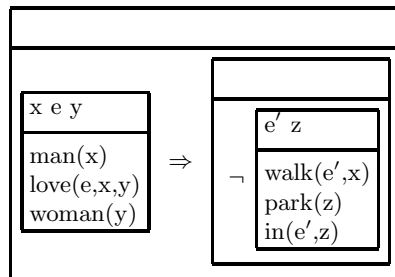


Fig. 1. DRS for *No man who loves a woman walks in a park*

DRSs can be given a direct model-theoretic interpretation [8,3,4]. Alternatively, the standard DRS language can be translated into first-order logic [14,15,16]. For instance, the DRS in Fig. 1 can be loosely paraphrased in first-order logic as *for all x, y and e , if x is a man, y is a woman, and x and y are part of a love event e , then it is not the case that there is a z and e' such that z is a park and x is involved in a walking event e' located in z* . We will make use of a translation function when mapping our proposed DRS language into the two-variable fragment of first-order logic.

3 Economical DRT

The standard DRS language of DRT is not decidable, because it has the same expressive power as first-order logic. The fragment of DRT that we introduce here, Economical DRT, is decidable, because we can show that the meaning representations of EDRT can be translated into FO². Let's first define the syntax of the meaning representation called EDRS (Economical Discourse Representation Structure):

Definition 1 (Syntax of EDRSs).

1. If $\{c_1 \cdots c_n\}$ is a non-empty set of EDRS-conditions, then $\langle \emptyset, \{c_1 \cdots c_n\} \rangle$ is an EDRS;
2. If u is a discourse referent, and $\{c_1 \cdots c_n\}$ is a non-empty set of EDRS-conditions, then $\langle \{u\}, \{c_1 \cdots c_n\} \rangle$ is an EDRS;
3. Nothing else is an EDRS.

Clause 1 says that an EDRS can have an empty domain. Clause 2 states that an EDRS has at most one discourse referent in its domain (unlike standard DRT). As we can see from this definition, EDRSs are mutually recursively defined with EDRS-conditions. So let's now consider the syntax of EDRS-conditions, where we use the variable B (possibly indexed) ranging over EDRSs:

Definition 2 (Syntax of EDRS-conditions).

1. If P is a one-place predicate symbol, and u is a discourse referent, then $P(u)$ is an EDRS-condition;
2. If R is a two-place predicate symbol, and u and u' are discourse referents, then $R(u, u')$ is an EDRS-condition;
3. If B is an EDRS, then $+B$ and $-B$ are EDRS-conditions;
4. If B_1 and B_2 are EDRSs, then $B_1 \Rightarrow B_2$ and $B_1 \vee B_2$ are EDRS-conditions;
5. Nothing else is an EDRS-condition.

Clause 1 and 2 define one-place and two-place relations, respectively. Unlike standard DRT, we don't need ternary or relations of higher arity. We come back to this issue when discussing events and thematic roles in the next section. Clause 3 defines complex conditions to record positive and negative information. Clause 4 defines implicational and disjunctive conditions, respectively. Most of these EDRS-conditions resemble those of standard DRT, with two exceptions: $-B$ marks negative information, and $+B$ marks positive information.

Note that the language of EDRT hosts only two discourse referents, named “1” and “2”. This restriction doesn’t mean that we can only name two different discourse referent in an EDRS — we may re-use discourse referents as often as we like, nested in sub-EDRSs. Examples below will illustrate this technique, such as the EDRS in Fig. 3.

Now that we know what the syntax of EDRSs looks like, let’s consider its semantics. The EDRS language is interpreted by translation to FO^2 with the aid of the function $[\cdot]^{fo2}$. This function is defined for discourse referents, EDRS-conditions, and EDRSs. The full translation is shown in Fig. 2. It always produces a formula of FO^2 , simply because it only yields at most two different variables (namely x and y).

Recall that in DRT, discourse representation structure serves as both the content and context. We have shown, with the help of the translation function in Fig. 2, how EDRT interprets *content*. With respect to *context*, we borrow some terminology of standard DRT to clarify the concepts *subordination*, *accessibility*, and introduce a new concept of *insertability*. Let’s first consider subordination of EDRSs (where we mean by c is a condition of B that B is a DRS $< D, C >$ and c is a member of C).

Definition 3 (EDRS Subordination). *Given an EDRS B , an EDRS B_1 subordinates an EDRS B_2 iff:*

1. $+B_2$ is an EDRS-condition of B_1 ,
2. $-B_2$ is an EDRS-condition of B_1 ,
3. $B \vee B_2$ is an EDRS-condition of B_1 ,
4. $B_2 \vee B$ is an EDRS-condition of B_1 ,
5. $B_2 \Rightarrow B$ is an EDRS-condition of B_1 ,
6. $B_1 \Rightarrow B_2$ is an EDRS-condition of B , or
7. B_1 subordinates B , and B subordinates B_2 .

Hence, subordination is recursively defined over EDRSs. We use it to define two more relations that play a crucial role for processing anaphoric pronouns in EDRT, which will be discussed in Section 6. The first is accessibility, which follows the definition from standard DRT. The second, new relation, is insertability.

$$\begin{aligned}
 [< \emptyset, \{c_1, \dots, c_n\} >]^{fo2} &= ([c_1]^{fo2} \wedge \dots \wedge [c_n]^{fo2}) \\
 [< \{u\}, \{c_1, \dots, c_n\} >]^{fo2} &= \exists [u]^{fo2} ([c_1]^{fo2} \wedge \dots \wedge [c_n]^{fo2}) \\
 [< \emptyset, \{c_1, \dots, c_n\} > \Rightarrow B]^{fo2} &= ([c_1]^{fo2} \wedge \dots \wedge [c_n]^{fo2}) \rightarrow [B]^{fo2} \\
 [< \{u\}, \{c_1, \dots, c_n\} > \Rightarrow B]^{fo2} &= \forall [u]^{fo2} ([c_1]^{fo2} \wedge \dots \wedge [c_n]^{fo2}) \rightarrow [B]^{fo2} \\
 [B_1 \vee B_2]^{fo2} &= ([B_1]^{fo2} \vee [B_2]^{fo2}) \\
 [+B]^{fo2} &= [B]^{fo2} \\
 [-B]^{fo2} &= \neg [B]^{fo2} \\
 [R(u, u')]^{fo2} &= R([u]^{fo2}, [u']^{fo2}) \\
 [P(u)]^{fo2} &= P([u]^{fo2}) \\
 [1]^{fo2} &= x \\
 [2]^{fo2} &= y
 \end{aligned}$$

Fig. 2. Translation from EDRS to the two-variable fragment of First-Order Logic

Definition 4 (Accessibility). *A discourse referent in an EDRS B is accessible from an EDRS B' iff:*

1. B subordinates B' , or
2. B equals B' .

Definition 5 (Insertability). *An EDRS can be inserted into another EDRS B (B is insertable) iff:*

1. There is no EDRS B' such that B' subordinates B , or
2. $+B$ is an EDRS-condition of B' and B' is insertable.

Accessibility is an important notion in DRT for establishing anaphoric links. Our EDRS language inherits the nice properties of accessibility of antecedents of pronouns, but also controls the use of pronouns by adding further restrictions on the structure of discourse. Insertability is the EDRT concept for conjoining sentences. It basically states at which level of discourse we can insert information coming from subsequent sentences. In the next section, which demonstrates the modelling of events and thematic roles in the EDRS language, we will illustrate this mechanism with various examples.

4 Events and Thematic Roles

Davidson suggested that sentences quantify over events and that adverbs and adjuncts essentially can be seen as modifiers of events, at least from a semantic point of view [17]. This is attractive from a modelling perspective, because the meaning of sentences with events can be captured by a simple first-order language, rather than a more complicated language based on higher-order logic. Davidson's view of "events introducing individuals", is adopted by standard DRT [3], shown by the DRS in Fig. 1.

Parsons, in turn, proposed a modification of Davidson's analysis [18]. He stipulated that not only modifiers but also the arguments of verbs should be viewed as predicates of events. The resulting framework is what is usually referred to as the *neo-Davidsonian* system [19]. It has two interesting consequences: (1) we can uniformly introduce *thematic roles* as relations between events and individuals; (2) verbs with optional arguments can be covered with a simpler signature of predicates.

Unlike standard DRT, we adopt the neo-Davidsonian representation of events. Thematic roles are modelled as two-place relations between events and other entities. We use the inventory of VerbNet to assign thematic roles to verbal arguments [20], such as *agent*, *patient* and *theme*. The neo-Davidsonian representation is central to our technique of re-using discourse referents in EDRT.

A first example illustrating this technique is shown in Fig. 3. Even though we only use two different discourse referents, five different entities are introduced. The nesting of the positive EDRS-conditions allows us to reuse discourse referents of what essentially would be a "flat" DRS in ordinary DRT. The use of a

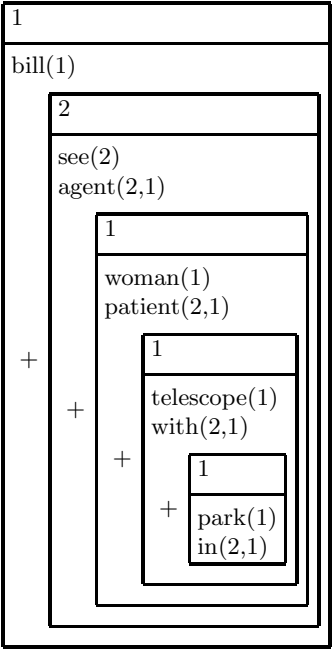


Fig. 3. EDRS for *Bill saw a woman with a telescope in the park*

neo-Davidsonian approach with thematic roles relating an event with a participating entity, rather than Davidson’s original proposal, ensures we don’t need more than two variables to analyse events. As a second example, consider again the DRS of standard DRT in Fig. 1 for *No man who loves a woman walks in a park*. In EDRT, this DRS is represented as shown in Fig. 4.

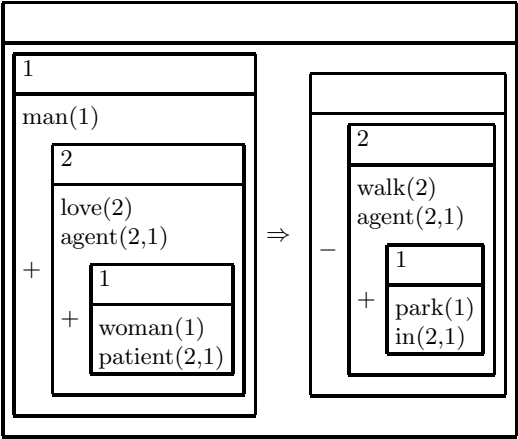


Fig. 4. EDRS for *No man who loves a woman walks in a park*

Perhaps these examples suggest that we can chain any number of arguments and modifiers within this neo-Davidsonian framework employing only two discourse referents, using one discourse referent denoting an event, and re-using another repeatedly to represent each of the participants. This is not the case. To ensure an adequate meaning representation, at most one quantifying noun phrase can be involved in each event in EDRT. This is because any quantified noun phrase needs to outscope the discourse referent introduced for the event — otherwise it wouldn't predict the correct truth conditions. Hence, we need to impose grammar constraints on the use of quantifiers in any fragment of English that is built on the basis of EDRT. One way to do this is by setting up the syntax-semantic interface in such a way that quantified noun phrase can only occur in subject position.

But what exactly do we mean by quantified noun phrases? Put simply, these are all noun phrases that introduce a universal quantifier when translated into first-order logic, or in DRT terms, those noun phrases that introduce an implicational DRS condition. Among these are noun phrases such as *everyone* and *nobody*, and those headed by the determiners *every*, *all*, and *no*. The non-quantifying noun phrases comprise indefinite and definite noun phrases, which all introduce discourse referents interpreted as existential quantifiers.

5 Building Economical DRSs

To obtain a syntactic structure for a natural language expression we employ a categorial grammar with basic categories *n* (noun), *np* (noun phrase), *s* (sentence), *pp* (prepositional phrase), and *t* (text). The notation of slashes follows CCG, hence a functor category α/β is subcategorising for a category β on its right yielding a category α , and a functor category $\alpha\backslash\beta$ is subcategorising for a category β on its left to produce a category α .

Each category corresponds to a (typed) partial EDRS. Because we only have two different discourse referents and no constant terms, β -conversion with renaming of variables to overcome accidental capture of free variables is not needed — instead it is safe to use unification for substitution. We will do so with a two-place operator $(A \cdot B)$, simulating to λ -abstraction [21], where B is always an EDRS, and A a discourse referent or another dot operation. Partial EDRSs can also only contain at most two distinct discourse referents.

In this version of EDRT we only use four combinatory rules: forward application ($>$), backward application ($<$), forward composition ($\mathbf{B}>$), and backward composition ($\mathbf{B}<$). They are defined in Fig. 5.

$$\begin{array}{ccc}
 \frac{\alpha/\beta: (X \cdot Y)}{\alpha: Y} & \beta: X > & \frac{\beta: X}{\alpha: Y} \quad \frac{\alpha\backslash\beta: (X \cdot Y)}{\alpha: Y} < \\
 \\
 \frac{\alpha/\beta: (X \cdot Y) \quad \beta/\gamma: (Z \cdot X)}{\alpha/\gamma: (Z \cdot Y)} > \mathbf{B} & & \frac{\beta\backslash\gamma: (Z \cdot X) \quad \alpha\backslash\beta: (X \cdot Y)}{\alpha\backslash\gamma: (Z \cdot Y)} < \mathbf{B}
 \end{array}$$

Fig. 5. Combinatory rules (α , β and γ range over CCG categories)

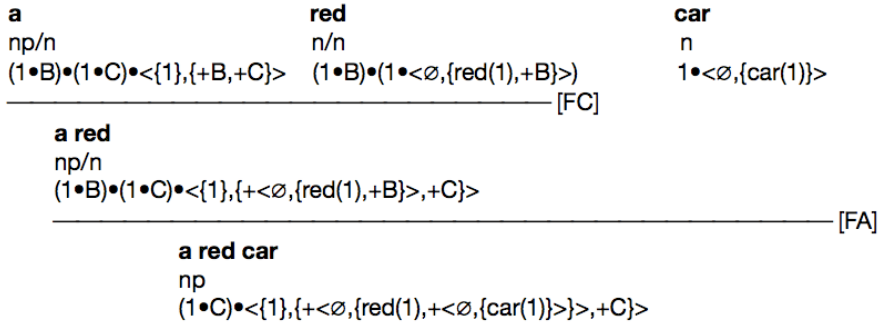


Fig. 6. CCG derivation for the noun phrase *a red car*, decorated with partial EDRSs

In Fig. 6 a CCG derivation, including EDRSs, for a simple noun phrase is shown, illustrating the combinatory rules forward composition and forward application. Fig. 7 shows several examples of lexical entries. Note that quantified determiners and pronouns are type-raised in the lexicon. This ensures that they only can occur in subject position, conform the discussion in Section 5. The entries for verbs in Fig. 7 illustrate how modifiers are dealt within the framework of a neo-Davidsonian event semantics.

6 Processing Pronouns

In this section we illustrate how the concept of insertability, introduced earlier in Section 3, determines the interpretation of pronouns. Pronouns introduce a free discourse referent “1”. Their interpretation depends on where they are inserted in the EDRS. Furthermore, the structure of an EDRS constrains insertability possibilities, and thereby correctly predicts the possibility and impossibility of certain anaphoric links, as in standard DRT.

Let’s first exemplify insertability. Consider the two EDRSs in Fig. 8. Recall the definition of insertability given in Section 3. This implies that the DRS for *Mia saw a woman* has three insertable DRSs (the outermost DRS, and the two embedded DRSs prefixed by the + operator), and the DRS for *Mia didn’t see a woman* has only one insertable DRS (the outermost DRS, the embedded DRS prefixed by + is blocked because it is not part of an insertable EDRS).

These insertability constraints predict that the third-person pronoun in the subsequent sentence *She smiled* can refer to either Mia or the woman she saw in the first case, but only to Mia in the second case. These possibilities are summarised as follows (where * marks an uninterpretable sentence):

- (1a) Mia^i saw a^j woman. She_i smiled.
- (1b) Mia^i saw a^j woman. She_j smiled.
- (2a) Mia^i didn’t see a^j woman. She_i smiled.
- (2b) Mia^i didn’t see a^j woman. * She_j smiled.

Cat	Partial EDRS	Entry
n	$(1 \cdot \boxed{\begin{array}{c} \\ \hline \text{car}(1) \end{array}})$	car
n/n	$((1 \cdot B) \cdot (1 \cdot \boxed{\begin{array}{c} \\ \hline \text{big}(1) \\ +B \end{array}}))$	big
np	$((1 \cdot B) \cdot \boxed{\begin{array}{c} 1 \\ \hline \text{person}(1) \\ +B \end{array}})$	someone
s/(s\ np)	$((1 \cdot B) \cdot \boxed{\begin{array}{c} \\ \hline \text{female}(1) \\ +B \end{array}} \cdot C) \cdot C$	she
np/n	$(1 \cdot C) \cdot (1 \cdot B) \cdot \boxed{\begin{array}{c} 1 \\ \hline +C \\ +B \end{array}}$	a
(s/(s\ np))/n	$(1 \cdot C) \cdot (((1 \cdot B) \cdot \boxed{\begin{array}{c} 1 \\ \hline +C \end{array}} \Rightarrow B) \cdot D) \cdot D$	every
s\ np	$((1 \cdot \boxed{\begin{array}{c} 2 \\ \hline \text{walk}(2) \\ \text{theme}(2,1) \\ +B \end{array}}) \cdot C) \cdot ((2 \cdot B) \cdot C)$	walks
(s\ np)/np	$((1 \cdot \boxed{\begin{array}{c} \\ \hline \text{theme}(2,1) \\ +D \end{array}}) \cdot B) \cdot (((1 \cdot \boxed{\begin{array}{c} 2 \\ \hline \text{see}(2) \\ \text{agent}(2,1) \\ +B \end{array}}) \cdot C) \cdot ((2 \cdot D) \cdot C))$	saw
pp	$(2 \cdot \boxed{\begin{array}{c} 1 \\ \hline \text{london}(1) \\ \text{to}(2,1) \end{array}})$	to London
t\ s	$((2 \cdot \boxed{\begin{array}{c} \\ \hline \text{event}(2) \end{array}}) \cdot B) \cdot B$.

Fig. 7. Mapping of CCG categories to EDRSs

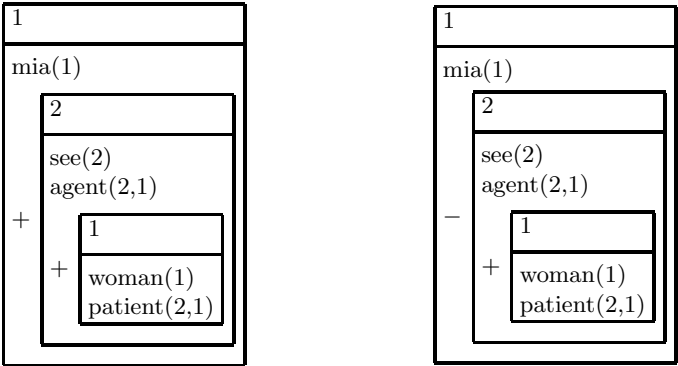


Fig. 8. EDRSs for *Mia saw a woman* (left) and *Mia didn't see a woman* (right)

These examples show how different insertability possibilities allow a pronoun to have several possible antecedents in EDRT. For instance, in Example (1a) above, the DRS for the second sentence “She smiled” can be inserted in the outermost DRS, resulting in an interpretation where the pronoun refers to Mia. Alternatively, the DRS containing the pronoun can be inserted in the deepest embedded `+`DRS, thereby establishing an anaphoric link between the third-person pronoun and “a woman”. Both possibilities are shown in Fig. 9.

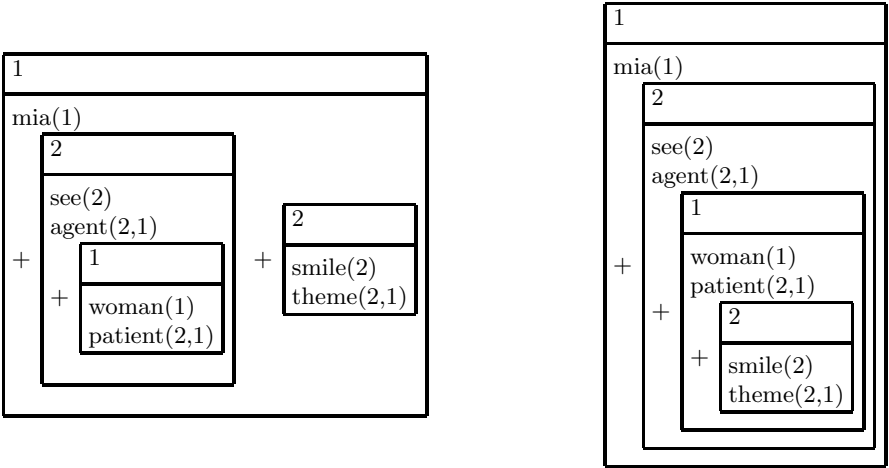


Fig. 9. Possible EDRSs for the discourse *Mia saw a woman. She smiled.*

The general mechanism for processing pronouns imposes a couple of limitations on the use of pronouns by EDRT. A simple sentence can have at most one pronoun referring back to antecedents introduced in the discourse processed so far. It is easy to see why: we can have at most two free discourse referents.

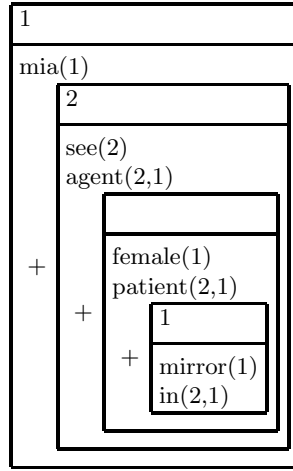


Fig. 10. EDRS for *Mia saw herself in the mirror*

But one discourse referent is already taken to represent the event introduced by the clause in which the pronoun occurs. That leaves one for any pronoun. This also suggest that pronouns need to be in subject position. Because if they would occur in object position, they would be interpreted as reflexive pronouns, referring to the subject of the sentential clause (Fig. 10).

If we assume that when processing a text in which sentences are interpreted incrementally, then we need to say something about how an EDRS constructed for a part of a certain text is combined with an EDRS of a subsequent sentence of this text. Put differently, we need to define conjoining of sentences. In ordinary DRT this is often done by merge reduction, a process in which the domains and conditions of two DRSs are taken together forming a new DRS. In EDRT no new machinery is needed for conjoining sentences, and merging of EDRSs is performed with the aid of the unary $+$ operator, which is already part of the EDRS language. A first definition of conjoining is the following:

Definition 6 (Conjoining). *Let B_1 and B_2 be EDRSs, and $B = \langle D, C \rangle$ an insertable EDRS in B_1 . Conjoining B_1 and B_2 yields the EDRS B_1 with B replaced by $\langle D, C \cup +B_2 \rangle$.*

This definition does essentially what is shown in Fig. 9. However, it is non-deterministic, because an EDRS can have several insertable sub-EDRSs, or “landing sites”. As a consequence, it could introduce spurious ambiguities. It actually does so for the DRS on the left-hand side in Fig. 8, since there are three possible locations for insertion: the outermost EDRS, the intermediate EDRS, and the deepest nested EDRS. The first possibility of insertion is shown on the left in Fig. 9, and the third possibility on the right. The second possibility would correspond to an interpretation equivalent to that of the first.

In order to remove these unwanted ambiguities, we redefine conjoining while distinguishing between open and closed EDRSs. A closed EDRS is an EDRS without free discourse referents (capturing the meaning of a sentence without pronouns), and an open EDRS is an EDRS with a free discourse referent (capturing the meaning of a sentence with a pronoun).

Let's first look at the case for closed EDRSs. Here we can constrain the landing site to be the outermost level of the EDRS (which is by definition insertable) representing the discourse processed so far. This is done in the following revised definition for conjoining:

Definition 7 (Conjoining for closed EDRSs). *Let B_1 and B_2 be EDRSs, such that $B_1 = \langle D, C \rangle$ and B_2 is closed. Conjoining B_1 and B_2 yields the EDRS $\langle D, C \cup +B_2 \rangle$.*

Now consider the case for open EDRSs. We call an open EDRS with a free variable u an “EDRS open for u ”. For an EDRS open for u , we can constrain the landing site to any insertable EDRS with a domain $\{u\}$. This yields the following definition:

Definition 8 (Conjoining for open EDRSs). *Let B_1 and B_2 be EDRSs, and $B = \langle \{u\}, C \rangle$ an insertable EDRS in B_1 , and B_2 an EDRS open for u . Conjoining B_1 and B_2 yields the EDRS B_1 with B replaced by $\langle \{u\}, C \cup +B_2 \rangle$.*

What about anaphoric phenomena related to pronouns, such as names and definite noun phrases? In EDRT, proper names and nouns headed by the definite article are dealt with in a similar way to indefinite noun phrases. Like indefinites, they introduce a discourse referent *in situ*. Unlike indefinites, they trigger a uniqueness presupposition, which is accommodated as part of the background knowledge. Uniqueness statements are formulated in first-order logic. This can be done with not more than two variables, so we're not leaving the two-variable fragment. Basic lexical knowledge can also be formulated in this way. Consider for instance the background knowledge computed for the EDRS of Fig. 10:

$$\begin{array}{ll}
 \forall x(\text{mia}(x) \rightarrow \text{person}(x)) & \forall x(\text{artifact}(x) \rightarrow \neg \text{event}(x)) \\
 \forall x(\text{mirror}(x) \rightarrow \text{artifact}(x)) & \forall x(\text{artifact}(x) \rightarrow \neg \text{person}(x)) \\
 \forall x(\text{see}(x) \rightarrow \text{event}(x)) & \forall x(\text{event}(x) \rightarrow \neg \text{person}(x)) \\
 \forall x \forall y((\text{mia}(x) \wedge \text{mia}(y)) \rightarrow x=y) & \\
 \forall x \forall y((\text{mirror}(x) \wedge \text{mirror}(y)) \rightarrow x=y) &
 \end{array}$$

7 Conclusion

The presented DRT fragment is of course far more restricted than the full blown version of DRT (for one, EDRT cannot represent donkey sentences). Nevertheless, we hope to have shown that the two-variable fragment has potential for controlled natural language applications. For instance, the use of neo-Davidsonian event-style semantics has no restrictions on the number of modifiers, even though it does on pronouns or universal quantified noun phrases.

Several questions remain to be resolved, most of them of theoretical nature. Some of them concern the translation from EDRS to the two-variable fragment of first-order logic. Is this translation meaning preserving? (We assume so, but we haven't given a proof.) Is there a translation from an arbitrary formula of FO^2 to EDRS? (If there isn't, EDRS would be a fragment of FO^2 .)

Further future work could address the integration of certain plural phrases in FO^2 , or in an extension of the EDRS language that still falls within a decidable fragment. And finally, the grammar presented in this article could be extended with a particular real-world application in mind. This would require a grammar covering more syntactic constructions, and be an ideal test for the proof-of-concept of using "semantically controlled" languages.

References

1. Mortimer, M.: On languages with two variables. *Zeitschrift für Math. Logik und Grundlagen der Mathematik* 21, 135–140 (1975)
2. de Nivelle, H., Pratt-Hartmann, I.: A resolution-based decision procedure for the two-variable fragment with equality. In: *IJCAR*, pp. 211–225 (2001)
3. Kamp, H., Reyle, U.: *From Discourse to Logic; An Introduction to Modeltheoretic Semantics of Natural Language, Formal Logic and DRT*. Kluwer, Dordrecht (1993)
4. van Eijck, J., Kamp, H.: Representing Discourse in Context. In: van Benthem, J., ter Meulen, A. (eds.) *Handbook of Logic and Language*, pp. 179–240. Elsevier, MIT (1997)
5. Pratt-Hartmann, I.: A two-variable fragment of english. *Journal of Logic, Language and Information* 12(1), 13–45 (2003)
6. Steedman, M.: *The Syntactic Process*. The MIT Press, Cambridge (2001)
7. Hockenmaier, J.: *Data and Models for Statistical Parsing with Combinatory Categorical Grammar*. PhD thesis, Univ. of Edinburgh (2003)
8. Kamp, H.: A Theory of Truth and Semantic Representation. In: Groenendijk, J., Janssen, T.M., Stokhof, M. (eds.) *Formal Methods in the Study of Language*, pp. 277–322. Mathematical Centre, Amsterdam (1981)
9. Klein, E.: *VP Ellipsis in DR Theory*. *Studies in Discourse Representation Theory and the Theory of Generalised Quantifiers* (1987)
10. van der Sandt, R.: Presupposition Projection as Anaphora Resolution. *Journal of Semantics* 9, 333–377 (1992)
11. Asher, N.: *Reference to Abstract Objects in Discourse*. Kluwer Academic Publishers, Dordrecht (1993)
12. Geurts, B.: *Presuppositions and Pronouns*. Elsevier, London (1999)
13. Kadmon, N.: *Formal Pragmatics*. Blackwell, Malden (2001)
14. Muskens, R.: Combining Montague Semantics and Discourse Representation. *Linguistics and Philosophy* 19, 143–186 (1996)
15. Blackburn, P., Bos, J., Kohlhase, M., de Nivelle, H.: Inference and Computational Semantics. In: Bunt, H., Muskens, R., Thijsse, E. (eds.) *Computing Meaning*, vol. 2, pp. 11–28. Kluwer, Dordrecht (2001)
16. Bos, J.: Computational Semantics in Discourse: Underspecification, Resolution, and Inference. *Journal of Logic, Language and Information* 13(2), 139–157 (2004)
17. Davidson, D.: The logical form of action sentences. In: Rescher, N. (ed.) *The Logic of Decision and Action*, pp. 81–95. University of Pittsburgh Press, Pittsburgh (1967)

18. Parsons, T.: Modifiers and quantifiers in natural language. *Canadian Journal of Philosophy* 6, 29–60 (1980)
19. Dowty, D.: On the semantic content of the notion “thematic role”. In: *Properties, Types, and Meanings*, vol. 2, pp. 69–129. Kluwer, Dordrecht (1989)
20. Kipper, K., Korhonen, A., Ryant, N., Palmer, M.: A large-scale classification of English verbs. *Language Resources and Evaluation* 42(1), 21–40 (2008)
21. Pereira, F., Shieber, S.: *Prolog and Natural Language Analysis*. CSLI Lecture Notes 10. Chicago University Press, Stanford (1987)

Controlled English Ontology-Based Data Access

Camilo Thorne and Diego Calvanese

KRDB Research Centre

Free University of Bozen-Bolzano

Via della Mostra, 4, 39100 Bolzano, Italy

{calvanese, cthorne}@inf.unibz.it

Abstract. As it is well-known, querying and managing structured data in natural language is a challenging task due to its ambiguity (syntactic and semantic) and its expressiveness. On the other hand, querying, e.g., a relational database or an ontology-based data access system is a well-defined and unambiguous task, namely, the task of evaluating a formal query (e.g., an SQL query) of a limited expressiveness over such database. However these formal query languages may be difficult to learn and use for the casual user and ambiguity may compromise the interface. To bridge this gap, the use of controlled language interfaces has been proposed. As a measure of their efficiency for data access, we propose to consider *data complexity*, which is the complexity of query evaluation measured *in the size of the data*. We study a family of controlled languages that express several fragments of OWL, ranging from tractable (**LogSpace** and **PTime**) to intractable (**coNP-hard**) in data complexity, singling out which constructs give rise to each computational property.

1 Introduction

Controlled languages (CLs) are subsets of natural language (NL) with minimal ambiguity tailored to fulfill data management tasks. Among these tasks, an important one is to provide “lightweight” front-end languages for non-expert users of information systems [30,6,7]. Data in information systems such as relational databases (DBs) is inserted, updated, deleted and queried using expressive formal query and data management languages such as, e.g., SQL, which are difficult to handle (let alone learn) for casual users. Language technologies have endeavoured to fill this gap by proposing wide coverage NL interfaces (NLI) that allow the user to use, say, everyday English, at the cost of system accuracy [2]. This is because NL is ridden with ambiguity (while formal languages are not) and thus NLIs have to choose between using complex heuristics or blowing up the translation process.

CL interfaces constitute a trade-off between the intuitive appeal of NLIs and the rigor of formal data query and management languages. Typically, CL expressions are translated symbolically into expressions belonging either to those formal languages or to some intermediate formalism (or logic) conveying a (formal) representation of its meaning (MR). Many different techniques, ranging from shallow (e.g., finite-state transducers) to deep (e.g., parsing), are used for defining such translations. Deep translations, in particular, have the property of inducing absolute accuracy without any significant loss

in translation efficiency. Indeed, deep translations are compositional, viz., the MR of a complete utterance is a function of the MRs of its (syntactic) constituents, i.e., semantics mirrors syntax. Hence, no information is lost in the translation [14]. When, in addition, the CL is context-free, compositional translations can be computed quite efficiently, i.e., we “compile” the CL in time polynomial in the length $|s|$ of the input string s .

Recently [6,19,25,27,26,28], CL interfaces to ontology-based data access systems (OBDASs) centered mostly around the W3C standard ontology language OWL¹ (Web Ontology Language) [11,16] have been proposed. They have given rise to a number of applications and implementations [6,19], among which ACE-OWL [11,16], which maps to OWL DL and fragments of it. The formal underpinning of OWL DL is provided by description logics (DLs) [3,13].

Given that OBDASs may contain large amounts of data, we are interested in knowing whether querying such systems through CL interfaces scales to data. An OBDAS can be modelled by a DL *knowledge base* (KB), whose information is accessed through queries belonging to some fragment of SQL. A fragment that is considered sufficiently expressive but still computationally manageable (since query answering is decidable in significant cases) is that of conjunctive queries [1]. The basic problem in this setting is (*conjunctive*) *query answering* (QA) [8], which is a form of logical entailment [3]. To evaluate the efficiency of querying through CL interfaces, we focus on the so-called *data complexity* of QA, i.e., the computational complexity of the problem measured in terms of the size of the data only [33]. Crucially, we want to know whether CL interfaces give way to *tractable* or *intractable* QA and, more in general, whether they affect positively or negatively the performance of OBDASs.

Tractable data complexity (viz., when the data complexity of QA is in **PTime**, that is, can be decided in at most polynomial time in the size of the data) indicates good scalability to data of query answering and CL interfaces. Intractable data complexity (viz., when the data complexity of QA is at least **coNP**-hard, hence beyond **PTime**) indicates, on the other hand, low scalability. The data complexity of query answering in OWL is known to be intractable [21]. Hence, CLs like ACE-OWL do not scale to data, although they contain fragments that do. This computational behaviour depends on the language constructs they cover. It would be of interest, therefore, for CL designers working with OBDASs to know which NL constructs (and in which combination) give rise to different computational properties.

In this paper we pinpoint some of the English constructs and a fortiori of ACE, that give rise to these computational properties by studying a space of CLs that translate into several fragments of OWL. In doing so, we make the following contributions:

- We study DL-English, a CL that maps into \mathcal{ALCC} , an intractable DL contained in OWL that is the smallest DL closed under boolean operations on concepts [3].
- By constraining DL-English constituents, we define a family of fragments thereof, viz., the $\{\text{IS-A}_i\}_{i \in [0,7]}$ family, that are either (i) *minimal* w.r.t. intractability or (ii) *maximal* w.r.t. tractability, mirroring Pratt and Third in [23] (cf. also [29]).
- We pinpoint the NL/CL constructs that are responsible for their different computational properties, in particular, for maximal tractability and minimal intractability.

¹ <http://www.w3.org/TR/owl-ref/>

2 Ontologies and Query Answering

2.1 Ontologies, Knowledge Bases and Queries

In an OBDAS, an ontology provides a conceptual view on the data stored in a relational database. OBDASs are formally underpinned by DL knowledge bases [8,3]. DLs are logics which structure the domain of discourse in terms of concepts (representing classes, i.e., sets of objects) and roles (representing binary relations between objects). We are interested in DLs of different expressiveness, viz., DL \mathcal{ALCT} and its fragments (such as $DL\text{-}Lite$ [8]). In \mathcal{ALCT} , *concepts* C and *roles* R are formed according to the following syntax:

$$R \rightarrow P \mid P^- \qquad C \rightarrow \top \mid A \mid \exists R:C \mid \neg C \mid C \sqcap C$$

where A stands for a concept name (a unary predicate), P for a role name (a binary predicate), and P^- for its inverse. We can enrich the set of \mathcal{ALCT} concepts, modulo the following (explicit) definitions: (i) $\forall R:C := \neg \exists R:\neg C$, (ii) $C \sqcup C' := \neg(\neg C \sqcap \neg C')$, (iii) $\perp := \neg \top$, and (iv) $\exists R := \exists R:\top$.

In a DL ontology \mathcal{O} , intensional knowledge is specified by means of a set of (concept inclusion) *assertions* of the form $C_l \sqsubseteq C_r$, stating inclusion (or IS-A) between the instances of the concept C_l on the *left* and those of the concept C_r on the *right*. In \mathcal{ALCT} , C_l and C_r may be arbitrary concepts, while fragments of \mathcal{ALCT} , such as $DL\text{-}Lite$ [8], can be obtained by suitably restricting the syntax for C_l and for C_r . A *database* (DB), expressing extensional knowledge, is a finite set \mathcal{D} of unary and binary ground atoms of the form $A(c)$, $P(c, c')$. A *knowledge base* (KB) is a pair $\langle \mathcal{O}, \mathcal{D} \rangle$, where \mathcal{O} is an ontology and \mathcal{D} a DB.

The semantics of DL concepts, assertions, ontologies, and KBs is specified by considering FO *interpretations* $\mathcal{I} := \langle \Delta, \cdot^{\mathcal{I}} \rangle$, where Δ is the *domain*, a possibly countably infinite set of constants, and $\cdot^{\mathcal{I}}$ is an *interpretation* function that maps (i) each concept name A to a subset of the domain, (ii) each role name P to a binary relation over the domain, and (iii) each constants c to itself. It can be extended to complex concepts C and roles R by structural recursion, as shown in Table 1.

An interpretation \mathcal{I} is said to be a *model* of an inclusion assertion $C_l \sqsubseteq C_r$, in symbols $\mathcal{I} \models C_l \sqsubseteq C_r$, if $C_l^{\mathcal{I}} \subseteq C_r^{\mathcal{I}}$. Similarly, \mathcal{I} is said to be a model of a ground atom $A(c)$ (or $P(c, c')$), in symbols $\mathcal{I} \models A(c)$ (resp. $\mathcal{I} \models P(c, c')$), if $c \in A^{\mathcal{I}}$ (resp. $\langle c, c' \rangle \in P^{\mathcal{I}}$). It is said to be a *model* of an ontology \mathcal{O} , in symbols $\mathcal{I} \models \mathcal{O}$ (resp. a DB \mathcal{D} , in symbols $\mathcal{I} \models \mathcal{D}$) if it is a model of all of \mathcal{O} 's assertions (resp. \mathcal{D} 's ground atoms). It is said to be a *model* of a KB $\langle \mathcal{O}, \mathcal{D} \rangle$, in symbols $\mathcal{I} \models \langle \mathcal{O}, \mathcal{D} \rangle$, if it is a model of \mathcal{O} and \mathcal{D} . Two concepts C and C' are said to be *equivalent* iff, for all \mathcal{I} , $C^{\mathcal{I}} = C'^{\mathcal{I}}$. Two assertions α and α' are said to be *equivalent* iff, for all \mathcal{I} , $\mathcal{I} \models \alpha$ iff $\mathcal{I} \models \alpha'$.

We use queries to retrieve information from KBs. As query languages, we consider *conjunctive queries* (CQs) [1], i.e., SELECT-PROJECT-JOIN SQL queries and *tree shaped conjunctive queries* (TCQs) [21], which are those CQs built using only unary and binary relations and that are tree-isomorphic.

A *conjunctive query* (CQ) q is an expression of the form

$$q(\bar{x}) \leftarrow \Phi(\bar{x}, \bar{y}),$$

Table 1. Semantics of the DL \mathcal{ALCC}

Syntax	Semantics
c	$c^{\mathcal{I}} := c$
A	$A^{\mathcal{I}} \subseteq \Delta$
\top	$\top^{\mathcal{I}} := \Delta$
$\exists R:C$	$(\exists R:C)^{\mathcal{I}} := \{d \mid \text{exists } d' \text{ s.t. } \langle d, d' \rangle \in R^{\mathcal{I}} \text{ and } d' \in C^{\mathcal{I}}\}$
$\neg C$	$(\neg C)^{\mathcal{I}} := \Delta \setminus C^{\mathcal{I}}$
$C \sqcap C'$	$(C \sqcap C')^{\mathcal{I}} := C^{\mathcal{I}} \cap C'^{\mathcal{I}}$
P	$P^{\mathcal{I}} \subseteq \Delta \times \Delta$
P^-	$(P^-)^{\mathcal{I}} := \{\langle d, d' \rangle \mid \langle d', d \rangle \in P^{\mathcal{I}}\}$
$C_l \sqsubseteq C_r$	$\mathcal{I} \models C_l \sqsubseteq C_r \text{ iff } C_l^{\mathcal{I}} \subseteq C_r^{\mathcal{I}}$
$A(c)$	$\mathcal{I} \models A(c) \text{ iff } c^{\mathcal{I}} \in A^{\mathcal{I}}$
$P(c, c')$	$\mathcal{I} \models P(c, c') \text{ iff } \langle c^{\mathcal{I}}, c'^{\mathcal{I}} \rangle \in P^{\mathcal{I}}$
\mathcal{O}	$\mathcal{I} \models \mathcal{O} \text{ iff for all } \alpha \in \mathcal{O}, \mathcal{I} \models \alpha$
\mathcal{D}	$\mathcal{I} \models \mathcal{D} \text{ iff for all } \alpha \in \mathcal{D}, \mathcal{I} \models \alpha$
$\langle \mathcal{O}, \mathcal{D} \rangle$	$\mathcal{I} \models \langle \mathcal{O}, \mathcal{D} \rangle \text{ iff } \mathcal{I} \models \mathcal{O} \text{ and } \mathcal{I} \models \mathcal{D}$

where $q(\bar{x})$ is the *head*, \bar{x} denotes a (finite) sequence of variables of *length* $|\bar{x}|$, the query's *distinguished variables*, and $\Phi(\bar{x}, \bar{y})$ is the *body*, a conjunction of FO relational atoms over the variables \bar{x} and \bar{y} .

Intuitively, non-distinguished variables combined with the relational conjunctions stand for relational DB table joins and selections, and distinguished variables for the information we want to project in the result: CQs thus constitute a declarative specification of a SQL SELECT-PROJECT-JOIN query result table [1].

A *tree-shaped condition* (TSC) of root z_i , for $i \in \mathbb{N}$, is a FO formula $\varphi(z_i)$ inductively defined as follows:

- every atom $A(z_i)$ or $R(z_i, z_{i+1})$ is a TSC,
- if $\varphi(z_{i+1})$ is a TSC, so is $R(z_i, z_{i+1}) \wedge \varphi(z_{i+1})$, and
- if $\varphi(z_i)$ and $\varphi'(z_i)$ are TSCs, so is $\varphi(z_i) \wedge \varphi'(z_i)$.

Note that, for all $i \in \mathbb{N}$, $z_i \neq z_{i+1}$, i.e., each time we apply the second rule we have to introduce a new variable. Furthermore, whenever $R(z_j, z_{j+1})$, occurs in $\varphi(z_i)$, for $j \geq i$, we say that z_{j+1} is *connected* to z_j .

Every TSC $\varphi(z_i)$ rooted in z_i and variables z_{i+1}, \dots, z_{i+m} can be (bijectively) mapped to a directed adorned tree T_φ : each variable z_j , for $j \in [i, i+m]$, gives rise to a node z_j , each atom $R(z_j, z_{j+1})$ to an edge $\langle z_j, z_{j+1} \rangle$ with tag R and each atom $A(z_j)$ to tag A over node z_j [3].

A *tree-shaped conjunctive query* (TCQ) q [21] is a CQ with one distinguished variable x and non distinguished variables y_1, \dots, y_n whose body $\Phi(x, y_1, \dots, y_n)$ can be written as a TSC $\varphi(x)$ of root x wherein y_1 is connected to x and each y_{i+1} is connected to y_i .

Let $q(\bar{x}) \leftarrow \Phi(\bar{x}, \bar{y})$ be a (T)CQ. A *grounded substitution* or *grounding* of q is a (not necessarily total) function $\sigma(\cdot)$ that maps the (free) variables $\bar{x} \cup \bar{y}$ of q to Δ . Groundings are extended to complex syntactic objects in the usual way. By $\sigma\theta$ we will denote the composition of grounding σ with grounding θ . We denote by $q\sigma$ the *grounding* of q 's

body by $\sigma(\cdot)$, i.e., $\Phi(\bar{x}, \bar{y})\sigma$. Notice that (T)CQ bodies are positive FO formulas. An interpretation \mathcal{I} is said to be a *model* of $q\sigma$ whenever $\mathcal{I} \models \Phi(\bar{x}, \bar{y})\sigma$ (following standard FO semantics, based on truth and satisfaction).

We say that a sequence \bar{c} of constants is an *answer* to a (T)CQ q with distinguished variables \bar{x} over a KB $\langle \mathcal{O}, \mathcal{D} \rangle$, whenever there exists a grounding σ , mapping \bar{x} to \bar{c} , s.t. $q\sigma$ is *logically entailed* by $\langle \mathcal{O}, \mathcal{D} \rangle$, viz., whenever, for all \mathcal{I} , $\mathcal{I} \models \langle \mathcal{O}, \mathcal{D} \rangle$ implies $\mathcal{I} \models q\sigma$; in symbols: $\langle \mathcal{O}, \mathcal{D} \rangle \models q\sigma$.

2.2 Query Answering and Data Complexity

We first recall some basic notions of complexity theory that we use in the following, and refer to [22] for more details. A *decision problem* P is a pair constituted by a set of *inputs* and a *question*, viz., a property to be verified for the inputs. Computational complexity refers to the (worst case) space and time resources an algorithm (i.e., a Turing machine) uses to solve a decision problem P , i.e., to determine whether an arbitrary input satisfies or not the problem. A *complexity class* or *computational property* is a class of decision problems. **LogSpace** denotes the class of problems P solvable by a deterministic algorithm using $\mathbf{O}(\log n)$ space, where n denotes the size of P 's input(s). **NLogSpace** denotes the class of problems solvable by a non-deterministic algorithm in, again, $\mathbf{O}(\log n)$ space. **PTime** denotes the class of problems solvable by a deterministic algorithm in $\mathbf{O}(p(n))$ time, where $p(\cdot)$ denotes a polynomial function. **coNP** denotes the class of problems whose complement is solvable by a non-deterministic algorithm in $\mathbf{O}(p(n))$ time. We have that **LogSpace** \subseteq **NLogSpace** \subseteq **PTime** \subseteq **coNP**. Higher complexity classes are defined similarly.

Given a class \mathbf{C} , a problem P is said to be **C-hard** whenever for each problem $P' \in \mathbf{C}$ there exists an algorithm, called *reduction*, that embeds P' into P and that runs in $\mathbf{O}(\log n)$ space in the size n of the input for P' . A **C-hard** problem P is as hard as any problem in \mathbf{C} , possibly harder, i.e., reductions make \mathbf{C} a complexity lower bound for P . If, in addition, \mathbf{C} is a complexity upper bound for P , i.e., if P can be shown to be in \mathbf{C} , P is said to be **C-complete**. Since log-space reductions are closed under composition, to show that a problem P is **C-hard** it suffices to reduce to P a problem P' that is already known to be **C-hard**.

Problems in **PTime** are said to be *tractable* and problems beyond **PTime** (e.g., **coNP-hard** problems) *intractable*, insofar as polynomial time traces the boundary up to which problems can be efficiently solved.

The *query answering* (QA) decision problem for (T)CQs and KBs is the entailment problem stated as follows:

- **Input:** a KB $\langle \mathcal{O}, \mathcal{D} \rangle$, a (T)CQ q with distinguished variables \bar{x} , and a sequence \bar{c} of $|\bar{x}|$ constants,
- **Question:** does there exist a grounded substitution $\sigma(\cdot)$ s.t. (i) $\sigma(\bar{x}) = \bar{c}$ and (ii) $\langle \mathcal{O}, \mathcal{D} \rangle \models q\sigma$?

We are interested in the *data complexity* of QA, namely, in its computational complexity when we consider \mathcal{D} as the only input of this problem [33]. The optimal data complexity for QA (w.r.t. (T)CQs) lies in **LogSpace**, which is roughly the complexity of

answering SQL queries over plain DBs²[1]. Such complexity is attained w.r.t. KBs from the *DL-Lite* fragment of the DL *ALCT* [8]. OWL and *ALCT* are, on the other hand, **coNP**-hard and **coNP**-complete, respectively [21]. Tractable data complexity implies OBDA scalability (to data).

3 Controlled Languages

CLs have been proposed as a means of overcoming the ambiguity problem inherent to NLI to information systems, but this is not the only knowledge representation (KR) or information management task CLs have been developed for. Other important tasks have been, e.g., (i) ontology authoring and (ii) declaring (and reasoning about) formal specifications. In such a setting, we are required to declare with CL complex structured information, such as OWL ontology assertions, or FO axiomatics. Expressive and compositional CLs such as ACE, ACE-OWL, PENG, SOS, etc., [25,15,27,26,28] have been developed with such purpose in mind. See Table 2 for an overview (CG stands for “categorical grammar”, DCG for “definite clause grammar” and the other acronyms for known English parsers and/or parser APIs such as GATE).

ACE-OWL, for instance, supports English verb phrase, nominal/noun and noun phrase-coordination, (full) negation, universal, existential, and counting quantifiers, and gap-filler dependencies induced by relative sentences (although quantification, mirroring quantification in OWL, is somewhat restricted [15,3]). Other constructs, inherently ambiguous in English, such as prepositional phrases (e.g., “John saw a man with the telescope”) are, on the other hand, not supported. Nor are adjectives (attributes are expressed by ad hoc transitive verbs, as is common in DLs), intransitive verbs (atomic concepts are expressed by means of common nouns) or query words (“who”, “which”, “what”, etc.), although ACE-OWL can be extended to an interrogative fragments that translates into TSQs [15].

Table 2. An overview of some CLs

CL (English)	Compositional	Maps to	Parser	Goal
ACE [12]	yes	FO	APE	KR/User specifications
ACE-OWL [15]	yes	OWL-DL	APE	Ontology authoring + querying
PENG [27]	yes	OWL-DL	ECOLE	Ontology authoring + querying
SOS [26]	N.A.	OWL-DL	N.A.	Ontology authoring + querying
CLCE [31]	yes	FO	compiler	KR
English Query [7]	N.A.	SQL	N.A.	DB querying/management
OWL-CNL [28]	yes	OWL-DL	DCG parser	Ontology authoring
Lite English[5]	yes	<i>DL-Lite</i>	CG parser	KR/Ontology authoring
λ -SQL [19]	yes	SQL	compiler	DB querying
nRQL [25]	yes	FO queries	DCG parser	Ontology querying
Rabbit [26]	no	OWL	GATE	Ontology authoring
ACE-PQL [6]	yes	PQL	DCG parser	Ontology querying

² Data complexity for DB query evaluation is actually in \mathbf{AC}^0 , a circuit-based complexity class that strictly includes **LogSpace** [1,22].

But if syntactic (i.e., parsing) complexity or ambiguity are not an issue for CLs, *semantic complexity*, however, is. (i) The formal model of a compositional translation for any fragment of NL, such as CLs, are formal semantics compositional translations $\tau(\cdot)$ [14,10] wherein NL utterances are mapped to formal logic MRs, typically higher order logic (HO) or first order logic (FO) MRs. To be more precise, they are mapped to FO extended with the types, the λ -abstraction, the application and β -normalization of the simply-typed lambda calculus [10]. (ii) Such a translation is semantics-preserving. Observations (i) and (ii) together imply that the NL fragment will inherit both the reasoning problems and computational properties of its MR logic. Such decision problems and computational properties have been called by Pratt & Third in [23] the *semantic complexity* of an NL fragment/CL. QA is one such decision problem and data complexity one such computational property.

As a result, whenever a (logical) reasoning problem is invoked in the back-end, the semantic complexity of the (otherwise ambiguity-free and efficiently processable) front-end CL can still affect the overall performance of a information system that, like OBDASs, relies heavily on FO reasoning. Modulo $\tau(\cdot)$ such impact can be studied *construct by construct*.

4 Expressing Ontologies with Controlled English

We make use of the standard formal semantics analysis for fragments of English as developed in [10,4,20]. We recall the so-called HO *typing rule*, which allows us to discard (when parsing) constituents that do not yield well-typed expressions. In particular, this rule fails whenever the types of the MRs acting as premises do not unify [10]:

$$app \frac{\Gamma \vdash u : \tau \quad \Gamma' \vdash v : \tau \rightarrow \tau'}{\Gamma, \Gamma' \vdash u(v) : \tau'}$$

where τ denotes an arbitrary type, u, v arbitrary HO expressions and Γ, Γ' typing contexts (sets of possibly empty variable typings). An HO expression is said to be *well-typed* whenever (i) it is typed and (ii) $\Gamma = \emptyset$.

We use *definite clause grammars* (DCGs), albeit written in phrase structure grammar style for simplicity, as grammar formalism for defining our CLs. Parsing amounts to walking or searching (depth or breadth first) a tree-shaped space of *parse states* $\langle \alpha, \varphi, \tau, \Gamma \rangle$, where α stands for a CL syntactic constituent, φ for its MR, τ for its type and Γ for its typing context. Transition between states is based on unification and type-checking [14], and is fired by the grammar rules. MRs are computed on the fly, during parsing. The states may also encode morphosyntactic information.

We consider, when defining our CLs, the following syntactic constituents. We consider as content word categories proper nouns (**Pns**), common nouns (**Ns**), transitive and intransitive verbs (**TVs** and **IVs**) and adjectives (**Adjs**). By way of function word categories, we consider determiners (**Dets**), (indeterminate) pronouns (**Pros**), relative pronouns (**Pros**), conjunctions (**Crds**) and negations (**Negs**). Finally, regarding non-primitive (whether recursive or not) constituents, we consider verb phrases (**VPs**), noun phrases (**NPs**) nominals (**Noms**) and complete sentences (**Ss**). For simplicity we have

barred out of the language, subordinate clauses, which are captured, basically, by **VPs** combining with **Relps** and **Noms**.

Given a constituent γ , parsing yields a unique DL MR up to *structural equivalence*. A concept C (resp. an assertion α) is said to be *structurally equivalent* to (a HO formula) $\varphi: e \rightarrow t$ iff C is equivalent to the DL concept φ . Following DL conventions [3], we associate (and map) the non-recursive word categories **N**, **Adj**, and **IV** to atomic concepts. Category **TV** is associated to role names. Recursive constituents, by contrast, are associated to arbitrary concepts.

We have shown elsewhere [5] how to express the optimal (for data complexity) fragment of \mathcal{ALCT} , $DL\text{-}Lite$. In this section we (i) express the **coNP**-complete (in data complexity for QA) \mathcal{ALCT} and then we show (ii) how to define and express in CL the maximal tractable fragments of \mathcal{ALCT} and the minimal intractable fragments of \mathcal{ALCT} . This will lead us to define in the first place the CL DL-English, expressing \mathcal{ALCT} , and by restricting DL-English's grammar the $IS\text{-}A_{i \in [0,7]}$ family of CLs and EL-English, viz., the minimal intractable and maximal tractable fragments of DL-English.

4.1 DL-English

Figure 1 introduces DL-English's grammar G_{DL} . For reasons of simplicity and space, we disregard morphology and polarity issues. We also omit specifying the (open) class of content words.

Lemma 1. *For all sentences S in $L(G_{DL})$, there exists an assertion α in \mathcal{ALCT} s.t. $\tau(S) \equiv \alpha$.*

Proof. In order to prove this lemma we will prove something more general, namely, that,

$$\text{for each } \mathbf{VP} \text{ or } \mathbf{Nom} \text{ constituent, there exists a concept } C \text{ in } \mathcal{ALCT} \quad (\dagger) \\ \text{s.t. } \tau(\mathbf{VP}) \equiv C \text{ (resp. } \tau(\mathbf{Nom}) \equiv C).$$

We prove (\dagger) by mutual induction on the length n (for $n \geq 1$) of G_{DL} derivations rooted in a **VP** or a **Nom**. We make use of unification to prune away undesired parse (sub)trees when walking through the space of parsing states, whenever (semantic) types and (morphosyntactic) features fail to unify. See Figures 2 and 3 for examples.

- ($n = 1$). We consider in this case the derivations

$$\mathbf{VP} \Rightarrow \mathbf{IV} \Rightarrow A, \quad \mathbf{Nom} \Rightarrow \text{is a } \mathbf{N} \Rightarrow \text{is a } A, \quad \mathbf{VP} \Rightarrow \text{is } \mathbf{Adj} \Rightarrow A,$$

which are in D_{DL} provided that we consider as part of our content lexicon the productions

$$\begin{aligned} \mathbf{IV} \rightarrow A, \tau(\mathbf{IV}) &:= A: e \rightarrow t, & \mathbf{N} \rightarrow A, \tau(\mathbf{N}) &:= A: e \rightarrow t, \\ \mathbf{Adj} \rightarrow A, \tau(\mathbf{Adj}) &:= A: e \rightarrow t, \end{aligned}$$

whence $\tau(\mathbf{TV}) = \tau(\mathbf{Nom}) \equiv A$, where A is an atomic concept.

(Prase structure rules)

S → **NP VP**

NP → **Det Nom**

VP → **TV NP**

VP → is a **Nom**

VP → is **TV** by **NP**

NP → **Pro Relp VP**

VP → is **Adj**

VP → **IV**

VP → is **Neg TV** by **NP**

NP → **Pro**

VP → does **Neg IV**

VP → is **Neg a Nom**

Nom → **Nom Relp VP**

Nom → **Adj Nom**

VP → is **Neg Adj**

VP → **VP Crd VP**

Nom → **Nom Crd Nom**

Nom → **N**

(Semantic actions)

$\tau(\mathbf{S}) := \tau(\mathbf{NP})(\tau(\mathbf{VP}))$

$\tau(\mathbf{VP}) := \tau(\mathbf{NP})(\tau(\mathbf{TV}))$

$\tau(\mathbf{VP}) := \tau(\mathbf{Crd})(\tau(\mathbf{VP}))(\tau(\mathbf{VP}))$

$\tau(\mathbf{VP}) := \tau(\mathbf{Neg})(\tau(\mathbf{NP})(\tau(\mathbf{TV})))$

$\tau(\mathbf{VP}) := \tau(\mathbf{Neg})(\tau(\mathbf{Adj}))$

$\tau(\mathbf{VP}) := \tau(\mathbf{Adj})$

$\tau(\mathbf{VP}) := \tau(\mathbf{Nom})$

$\tau(\mathbf{VP}) := \tau(\mathbf{Neg})(\tau(\mathbf{Nom}))$

$\tau(\mathbf{VP}) := \tau(\mathbf{Neg})(\tau(\mathbf{IV}))$

$\tau(\mathbf{NP}) := \tau(\mathbf{Pro})$

$\tau(\mathbf{VP}) := \tau(\mathbf{IV})$

$\tau(\mathbf{NP}) := \tau(\mathbf{Det})(\tau(\mathbf{Nom}))$

$\tau(\mathbf{NP}) := \tau(\mathbf{Pro})(\tau(\mathbf{Relp})(\tau(\mathbf{VP})))$

$\tau(\mathbf{Nom}) := \tau(\mathbf{Nom})(\tau(\mathbf{Relp})(\tau(\mathbf{VP})))$

$\tau(\mathbf{Nom}) := \tau(\mathbf{Crd})(\tau(\mathbf{Nom}))(\tau(\mathbf{Nom}))$

$\tau(\mathbf{Nom}) := \tau(\mathbf{Adj})(\tau(\mathbf{Nom}))$

$\tau(\mathbf{Nom}) := \tau(\mathbf{N})$

(Function lexicon)

Pro → anybody $\tau(\mathbf{Pro}) := \lambda C. \lambda C'. C \sqsubseteq C' \quad (e \rightarrow t) \rightarrow ((e \rightarrow t) \rightarrow t)$

Pro → somebody $\tau(\mathbf{Pro}) := \lambda R. \exists R \quad (e \rightarrow (e \rightarrow t)) \rightarrow (e \rightarrow t)$

Pro → nobody $\tau(\mathbf{Pro}) := \lambda C. \lambda C'. C \sqsubseteq \neg C' \quad (e \rightarrow t) \rightarrow ((e \rightarrow t) \rightarrow t)$

Pro → nobody $\tau(\mathbf{Pro}) := \lambda R. \neg \exists R \quad (e \rightarrow (e \rightarrow t)) \rightarrow (e \rightarrow t)$

Crd → and $\tau(\mathbf{Crd}) := \lambda C. \lambda C'. C \sqcap C' \quad (e \rightarrow t) \rightarrow ((e \rightarrow t) \rightarrow (e \rightarrow t))$

Crd → or $\tau(\mathbf{Crd}) := \lambda C. \lambda C'. C \sqcup C' \quad (e \rightarrow t) \rightarrow ((e \rightarrow t) \rightarrow (e \rightarrow t))$

Relp → who $\tau(\mathbf{Relp}) := \lambda C. C \quad (e \rightarrow t) \rightarrow (e \rightarrow t)$

Relp → who $\tau(\mathbf{Relp}) := \lambda C. \lambda C'. C : C' \quad (e \rightarrow t) \rightarrow ((e \rightarrow t) \rightarrow (e \rightarrow t))$

Neg → not $\tau(\mathbf{Neg}) := \lambda C. \neg C \quad (e \rightarrow t) \rightarrow (e \rightarrow t)$

Pro → only $\tau(\mathbf{Pro}) := \lambda C. \lambda R. \forall R: C \quad (e \rightarrow t) \rightarrow ((e \rightarrow (e \rightarrow t)) \rightarrow (e \rightarrow t))$

Pro → everybody $\tau(\mathbf{Pro}) := \lambda C. \top \sqsubseteq C \quad (e \rightarrow t) \rightarrow t$

Pro → nobody $\tau(\mathbf{Pro}) := \lambda C. C \sqsubseteq \perp \quad (e \rightarrow t) \rightarrow t$

Det → some $\tau(\mathbf{Det}) := \lambda C. \lambda R. \exists R: C \quad (e \rightarrow t) \rightarrow ((e \rightarrow (e \rightarrow t)) \rightarrow (e \rightarrow t))$

Det → every $\tau(\mathbf{Det}) := \lambda C. \lambda C'. C \sqsubseteq C' \quad (e \rightarrow t) \rightarrow ((e \rightarrow t) \rightarrow t)$

Det → no $\tau(\mathbf{Det}) := \lambda C. \lambda C'. C \sqsubseteq \neg C' \quad (e \rightarrow t) \rightarrow ((e \rightarrow t) \rightarrow t)$

Fig. 1. The grammar G_{DL} of DL-English

- $(n = k + 1)$. By induction hypothesis, for every derivation of length $i \leq k$ rooted in **VP** or **Nom**, there exists a concept C s.t.

$$\mathbf{VP} \Rightarrow_i \gamma \text{ and } \tau(\mathbf{VP}) \equiv C \quad \text{or} \quad \mathbf{Nom} \Rightarrow_i \gamma \text{ and } \tau(\mathbf{Nom}) \equiv C, \quad (\text{IH})$$

where γ stands for a component derived (in G_{DL}) from **VP** (resp. **Nom**) in $i \leq k$ steps. We want to prove that the property holds for $\mathbf{VP} \Rightarrow_{k+1} \gamma$ and $\mathbf{Nom} \Rightarrow_{k+1} \gamma$.

We have several cases to consider, namely as many as there are recursive rules for **VP** and **Nom** in G_{DL} .

- **Nom** \Rightarrow **Adj Nom** $\Rightarrow_k \gamma\gamma'$. By induction hypothesis, there exists a concept C' s.t. $\tau(\gamma') \equiv C'$. Now, **Adj** is a *qualificative* adjective and we know from G_{DL} that in this case **Adj** $\Rightarrow A$ with MR $\tau(\mathbf{Adj}) := \lambda D^{e \rightarrow t}.(A \sqcap D): (e \rightarrow t) \rightarrow (e \rightarrow t)$. Thus,

$$\begin{aligned} \tau(A\gamma') &=_{df} \lambda D^{e \rightarrow t}.(A \sqcap D)(\tau(\gamma')): e \rightarrow t \\ &=_{ih} \lambda D^{e \rightarrow t}.(A \sqcap D)(C'): e \rightarrow t \\ &\triangleright_{\beta} A \sqcap C': e \rightarrow t, \end{aligned}$$

and $A \sqcap C'$ is the concept we were looking for.

- **VP** \Rightarrow **TV NP** \Rightarrow **TV Det Nom** $\Rightarrow_{k-1} \gamma\gamma'\gamma''$. By induction hypothesis, there exists a concept C'' s.t. $\tau(\gamma'') \equiv C''$. We know that in G_{DL} **TV** $\Rightarrow P$, with $\tau(\mathbf{TV}) \equiv P$. There are only two possibilities for **Det**:

$$\mathbf{Det} \Rightarrow \text{only} \quad \text{or} \quad \mathbf{Det} \Rightarrow \text{some}.$$

Let us focus, w.l.o.g. on the former. We know that in such a case it holds that $\tau(\mathbf{Det}) := \lambda D^{e \rightarrow t}.\lambda R^{e \rightarrow (e \rightarrow t)}.(\forall R:D)(\tau(\gamma'')): (e \rightarrow (e \rightarrow t)) \rightarrow ((e \rightarrow t) \rightarrow (e \rightarrow t))$. Therefore,

$$\begin{aligned} \tau(P \text{ only } \gamma'') &=_{df} \lambda D^{e \rightarrow t}.\lambda R^{e \rightarrow (e \rightarrow t)}.(\forall R:D)(\tau(\gamma''))(\tau(P)): e \rightarrow t \\ &=_{ih} \lambda D^{e \rightarrow t}.\lambda R^{e \rightarrow (e \rightarrow t)}.(\forall R:D)(C'')(P): e \rightarrow t \\ &\triangleright_{\beta} \forall P:C'': e \rightarrow t, \end{aligned}$$

and, clearly, $\tau(\mathbf{VP}) \equiv \forall P:C''$. Notice that any other choice for **Det** would prevent any derivation of the whole constituent. For instance, while **Det** \Rightarrow every, constituent of (partial) MR $\tau(\mathbf{Det}) := \lambda D^{e \rightarrow t}.\lambda E^{e \rightarrow t}.D \sqsubseteq E: t$, we cannot apply $\lambda E^{e \rightarrow t}.C'' \sqsubseteq E: e \rightarrow t$ to $P: e \rightarrow (e \rightarrow t)$, due to our HO type system. See Figure 3.

- All the other cases are dealt with analogously.

We now turn to complete utterances, viz. to DL-English sentences. There are three different ways in which a sentence S can be generated in our CL, namely:

- **S** \Rightarrow **NP VP** \Rightarrow **Det Nom VP** $\Rightarrow_* \gamma\gamma'\gamma''$ (with $S = \gamma\gamma'\gamma''$). We know that $\tau(\gamma') \equiv C'$ and that $\tau(\gamma'') \equiv C''$. Due to the typing constraints, the only possibilities for **Det** are:

$$\mathbf{Det} \Rightarrow \text{every} \quad \text{or} \quad \mathbf{Det} \Rightarrow \text{no},$$

with MR $\tau(\mathbf{Det}) := \lambda D^{e \rightarrow t}.\lambda E^{e \rightarrow t}.D \sqsubseteq E: (e \rightarrow t) \rightarrow ((e \rightarrow t) \rightarrow t)$ (resp. $\tau(\mathbf{Det}) := \lambda D^{e \rightarrow t}.\lambda E^{e \rightarrow t}.D \sqsubseteq \neg E: (e \rightarrow t) \rightarrow ((e \rightarrow t) \rightarrow t)$). Hence,

$$\begin{aligned} \tau(\text{every}\gamma'\gamma'') &=_{df} \lambda D^{e \rightarrow t}.\lambda E^{e \rightarrow t}.D \sqsubseteq E(\tau(\gamma'))(\tau(\gamma'')): t \\ &=_{\dagger} \lambda D^{e \rightarrow t}.\lambda E^{e \rightarrow t}.D \sqsubseteq E(C')(C''): t \\ &\triangleright_{\beta} C' \sqsubseteq C'': t. \end{aligned}$$

- $S \Rightarrow \mathbf{NP VP} \Rightarrow \mathbf{Pro Relp VP VP} \Rightarrow_* \gamma\gamma'\gamma''\gamma'''$ (with $S = \gamma\gamma'\gamma''\gamma'''$). Similar argument.
- $S \Rightarrow \mathbf{NP VP} \Rightarrow \mathbf{Pro VP} \Rightarrow_* \gamma\gamma'$ (with $S = \gamma\gamma'$). Similar argument.

Therefore, for each S in $L(G_{DL})$ there exists an assertion α s.t. $\tau(S) \equiv \alpha$. This completes the proof. \square

The *negation normal form* (NNF) of an \mathcal{ALCT} concept C is defined by pushing negation down to atomic concepts using the De Morgan laws [3]. For every concept C in \mathcal{ALCT} there exists an equivalent \mathcal{ALCT} concept C' in NNF.

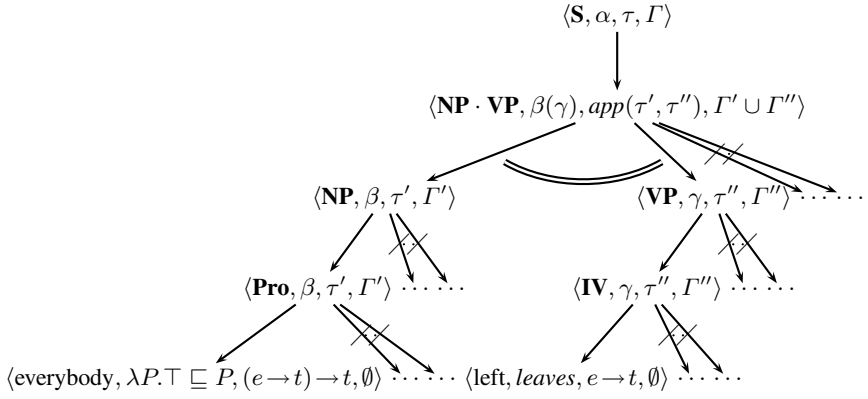


Fig. 2. A successful derivation for “everybody left”. The dots indicate transitions to failing states. The information propagated from leaves to root via unification yields, ultimately, $\langle \text{everybody left}, \top \sqsubseteq \text{leaves}, t, \emptyset \rangle$, i.e., a well-typed string and MR of type sentence (or t).

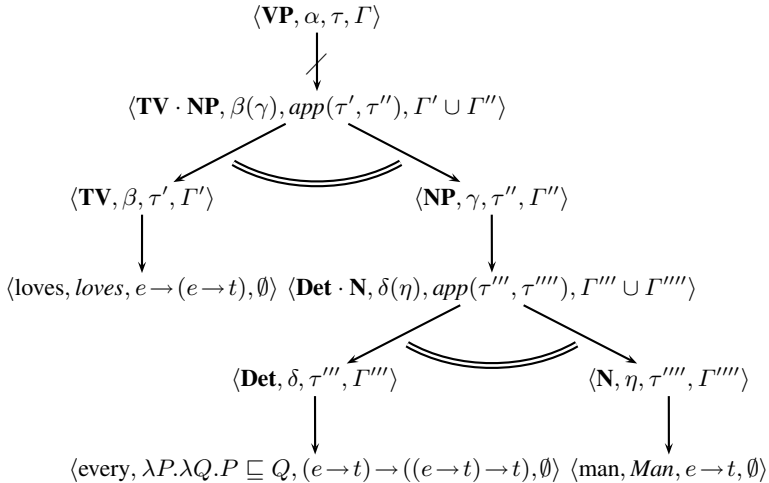


Fig. 3. A failed derivation for the VP “loves every man”, since $\text{app}(e \rightarrow (e \rightarrow t), e \rightarrow t) = \uparrow$, whence the string is not well-typed and is devoid of a (partial) MR

Lemma 2. *For each assertion α in \mathcal{ALCT} , there exists a sentence S in $L(G_{DL})$ s.t. (i) $\tau(S) \equiv \alpha'$, where α' is an \mathcal{ALCT} assertion, and (ii) α' is equivalent to α .*

Proof. Again, in order to prove this lemma, we prove a more general claim, namely, that

$$\text{for each concept } C \text{ in NNF, there exists either a } \mathbf{VP} \text{ or a } \mathbf{Nom} \quad (\dagger) \\ \text{s.t. } \tau(\mathbf{VP}) \equiv C' \text{ or } \tau(\mathbf{Nom}) \equiv C',$$

where C' is equivalent to C . This we prove by induction on C .

- (Basis). There are two cases to consider, given that C is in NNF.
 - $C := A$. Notice that A is already in NNF. Include in the function lexicon of G_{DL} the (terminal) production $\mathbf{N} \rightarrow A$ with lexical semantics $\tau(\mathbf{N}) := A: e \rightarrow t$. There are two possibilities:
 - * either $\mathbf{Nom} \Rightarrow \mathbf{N} \Rightarrow A$,
 - * or $\mathbf{VP} \Rightarrow$ is a $\mathbf{Nom} \Rightarrow$ is a $\mathbf{N} \Rightarrow$ is a A
 Notice, furthermore that we can express \top with the ad hoc \mathbf{N} “thing”, which is the usual DL convention [3].
 - $C := \neg A$. then $\mathbf{VP} \Rightarrow$ is \mathbf{Neg} a $\mathbf{Nom} \Rightarrow$ is not a $\mathbf{N} \Rightarrow$ is not a A , with $\tau(\mathbf{VP}) \equiv \neg A$, by the same argument as before, which is in NNF.
- (Inductive step). By inductive hypothesis we know that, for all subconcepts C' of C , there exists a component γ rooted in a \mathbf{VP} or \mathbf{Nom} s.t.:

$$\mathbf{VP} \Rightarrow_* \gamma \text{ and } \tau(\mathbf{VP}) \equiv C'' \quad \text{or} \quad \mathbf{Nom} \Rightarrow_* \gamma \text{ and } \tau(\mathbf{Nom}) \equiv C'', \quad (\text{IH})$$

where C'' is a concept equivalent to C' . This leaves two cases to consider, namely:

- $C := \exists P:C'$. By (IH), there exists a γ' s.t. either $\mathbf{VP} \Rightarrow_* \gamma'$ and $\tau(\mathbf{VP}) \equiv C''$, or $\mathbf{Nom} \Rightarrow_* \gamma'$, $\tau(\mathbf{Nom}) \equiv C''$, and C'' is equivalent to C' . This gives us two possibilities for $\exists P:C'$, namely:
 - * $\mathbf{VP} \Rightarrow \mathbf{TV NP} \Rightarrow \mathbf{TV Det Nom} \Rightarrow_* P$ some γ' , with $\tau(\mathbf{VP}) \equiv \exists P:C'$, which is equivalent to $\exists P:C$, or
 - * $\mathbf{VP} \Rightarrow \mathbf{TV NP} \Rightarrow \mathbf{TV Pro Relp VP} \Rightarrow_* P$ somebody who γ' , where $\tau(\mathbf{VP}) \equiv \exists P:C'$, which is equivalent to $\exists P:C$.
- $C := C' \sqcap C''$. Similar argument.

Let now $C \sqsubseteq C'$ be an \mathcal{ALC} assertion with C, C' in NNF. We can capture this assertion in one of two ways in DL-English:

- either $\mathbf{S} \Rightarrow_*$ every $\mathbf{Nom VP} \Rightarrow_*$ every $\gamma\gamma'$,
- or $\mathbf{S} \Rightarrow_*$ everybody who $\mathbf{VP VP} \Rightarrow_*$ everybody who $\gamma\gamma'$,

for some (two) components γ' and γ' whose existence is guaranteed by (\dagger) . Clearly, in both cases $\tau(\mathbf{S}) \equiv C'' \sqsubseteq C'''$. Moreover, it is evident that $C'' \sqsubseteq C'''$ is equivalent to $C \sqsubseteq C'$. This completes the proof. \square

From Lemmas 1 and 2, it follows immediately that DL-English expresses \mathcal{ALCT} (up to equivalence).

Theorem 1 (DL-English). *DL-English expresses \mathcal{ALCT} .*

Corollary 1. *QA for DL-English and (T)CQs is **coNP**-complete in data complexity.*

Example 1. Examples of sentences in DL-English (we spell out the MRs underneath) are:

No man who runs some business that does not make some money is shrewd.
 $Man \sqcap \exists run:(Business \sqcap \neg(\exists make:Money)) \sqsubseteq \neg Shrewd$ (1)

Nobody eats only apples.
 $\forall eats:Apple \sqsubseteq \perp$ (2)

Everybody sleeps.
 $\top \sqsubseteq Sleep$ (3)

Every married man has some wife.
 $Man \sqcap Married \sqsubseteq \exists has:Wife$ (4)

Anybody who has some car drives some new car or old car.
 $\exists has:Car \sqsubseteq \exists drives:((Car \sqcap New) \sqcup ((Car \sqcap Old)))$ (5)

4.2 The Family $\{IS-A_i\}_{i \in [0,7]}$ of CLs

We now turn to the computational properties of each of the constructs of DL-English *in isolation*. We do it by essentially restricting the kind of *right* (i.e., C_r) and *left* (i.e., C_l) concepts we may express. All utterances comply with the sentence patterns

“every $\gamma_l \gamma_r$ ” and “everybody who $\gamma_l \gamma_r$ ”.

The constituents γ_l and γ_r map to, respectively, left and right concepts, while sentences map to IS-A assertions of the form $C_l \sqsubseteq C_r$. We consider in this paper only 8 out of all possible combinations obtained by allowing in C_l and C_r some subset of the DL constructs in Table 3, giving rise to the family $\{IS-A_i\}_{i \in [0,7]}$ of CLs shown in Table 4. The basic kind of assertion they all express is IS-A among atomic concepts, viz., $A \sqsubseteq A'$, captured by IS-A₀. Notice that this can be easily achieved by, so to speak, merging the grammars from Table 3 expressing (in isolation) the C_l s and C_r s occurring in the DL assertions of Table 4 into a grammar for each CL IS-A_i.

Theorem 2 (IS-A_is). *For each $i \in [0,7]$ and each sentence S_i in IS-A_i, there exists an assertion α_i s.t. $\tau(S_i) \equiv \alpha_i$. Conversely, for each assertion α_i there exists a sentence S_i in IS-A_i s.t. $\tau(S_i) \equiv \alpha'_i$ and α'_i is equivalent to α_i .*

Proof. The theorem can be proved for each fragment in a manner analogous to DL-English. Basically, we consider two cases both on the “ \Rightarrow ” and the “ \Leftarrow ” directions of the proof, viz., a (mutual) induction on either (i) **Nom_l** and **VP_l** constituents or (ii) on the **Nom_r** and **VP_r** constituents for the if direction and a (mutual) induction over (i) a C_l or (ii) a C_r concept for the only if direction. With some routine adjustments to the specific syntax of the fragments and their MRs, we adapt each time the proof of Theorem 1.

Table 3. Expressing concepts C_f , for $f \in \{l, r\}$, and assertions $C_l \sqsubseteq C_r$, by restricting and subcategorizing rules in G_{DL}

$S \rightarrow \mathbf{NP}_l \mathbf{VP}_r \quad \mathbf{NP}_l \rightarrow \mathbf{Pro}_l \mathbf{Relp}_l \mathbf{VP}_l \quad \mathbf{NP}_l \rightarrow \mathbf{Det}_l \mathbf{Nom}_l$ $\mathbf{Pro}_l \rightarrow \text{anybody} \quad \mathbf{Relp}_l \rightarrow \text{who} \quad \mathbf{Det}_l \rightarrow \text{every}$		
Concept C_f	Constituent γ_f	Grammar Rules
$\exists P:A$	$\mathbf{TV} \text{ some } \mathbf{Nom}_f$ $\mathbf{TV} \text{ somebody who } \mathbf{VP}_f$	$\mathbf{VP}_f \rightarrow \text{is a } \mathbf{Nom}_f \mid \mathbf{IV} \mid \text{is } \mathbf{Adj}$ $\mathbf{Nom}_f \rightarrow \mathbf{N}$
$\exists P^-:A$	$\text{is } \mathbf{TV} \text{ by some } \mathbf{Nom}_f$ $\text{is } \mathbf{TV} \text{ by somebody who } \mathbf{VP}_f$	$\mathbf{VP}_f \rightarrow \text{is a } \mathbf{Nom}_f \mid \mathbf{IV}$ $\quad \mid \text{is } \mathbf{Adj} \mid \mathbf{TV} \mathbf{NP}_f$ $\mathbf{Nom}_f \rightarrow \mathbf{N}$ $\mathbf{NP}_f \rightarrow \mathbf{Det}_f \mathbf{Nom}_f$ $\quad \mid \mathbf{Pro}_f \mathbf{Relp}_f \mathbf{VP}_f$ $\mathbf{Det}_f \rightarrow \text{some}$ $\mathbf{Relp}_f \rightarrow \text{who}$ $\mathbf{Pro}_f \rightarrow \text{somebody}$
$\forall P:A$	$\mathbf{TV} \text{ only } \mathbf{VP}_f$ $\mathbf{TV} \text{ only who } \mathbf{VP}_f$	$\mathbf{VP}_f \rightarrow \text{is a } \mathbf{Nom}_f \mid \mathbf{IV}$ $\quad \mid \text{is } \mathbf{Adj} \mid \mathbf{TV} \mathbf{NP}_f$ $\mathbf{Nom}_f \rightarrow \mathbf{N}$ $\mathbf{NP}_f \rightarrow \mathbf{Det}_f \mathbf{Nom}_f$ $\quad \mid \mathbf{Pro}_f \mathbf{Relp}_f \mathbf{VP}_f$ $\mathbf{Det}_f \rightarrow \text{only}$ $\mathbf{Relp}_f \rightarrow \text{who}$ $\mathbf{Pro}_f \rightarrow \text{only}$
A	\mathbf{Nom}_f \mathbf{VP}_f	$\mathbf{VP}_f \rightarrow \text{is a } \mathbf{Nom}_f \mid \mathbf{IV} \mid \text{is } \mathbf{Adj}$ $\mathbf{Nom}_f \rightarrow \mathbf{N}$
$A_1 \sqcap \dots \sqcap A_n$	$\mathbf{Adj} \mathbf{Nom}_f$ $\mathbf{Nom}_f \text{ who } \mathbf{VP}_f$ $\mathbf{Nom}_f \text{ and } \mathbf{Nom}_f$ $\mathbf{VP}_f \text{ and } \mathbf{VP}_f$	$\mathbf{VP}_f \rightarrow \text{is a } \mathbf{Nom}_f \mid \mathbf{IV}$ $\quad \mid \text{is } \mathbf{Adj} \mid \mathbf{VP}_f \mathbf{Crd}_f \mathbf{VP}_f$ $\mathbf{Nom}_f \rightarrow \mathbf{N} \mid \mathbf{Adj} \mathbf{Nom}_f$ $\quad \mid \mathbf{Nom}_f \mathbf{Crd}_f \mathbf{Nom}_f$ $\quad \mid \mathbf{Nom}_f \mathbf{Relp}_f \mathbf{VP}_f$ $\mathbf{Relp}_f \rightarrow \text{who}$ $\mathbf{Crd}_f \rightarrow \text{and}$
$\exists P$	$\mathbf{TV} \text{ something}$ $\mathbf{TV} \text{ somebody}$	$\mathbf{VP}_f \rightarrow \mathbf{TV} \mathbf{Pro}_f$ $\mathbf{Pro}_f \rightarrow \text{somebody} \mid \text{something}$
$A_1 \sqcup \dots \sqcup A_n$	$\mathbf{VP}_f \text{ or } \mathbf{VP}_f$	$\mathbf{VP}_f \rightarrow \text{is a } \mathbf{Nom}_f \mid \mathbf{IV}$ $\quad \mid \text{is } \mathbf{Adj} \mid \mathbf{VP}_f \text{ and } \mathbf{VP}_f$ $\mathbf{Nom}_f \rightarrow \mathbf{N} \mid \mathbf{Nom}_f \text{ and } \mathbf{Nom}_f$
$\neg A$	$\text{is not } \mathbf{Adj}$ $\text{does not } \mathbf{IV}$ $\text{is not a } \mathbf{Nom}_f$	$\mathbf{Nom}_f \rightarrow \mathbf{N}$ $\mathbf{VP}_f \rightarrow \text{does not } \mathbf{IV}$ $\quad \mid \text{is not } \mathbf{Adj}$ $\quad \mid \text{is not a } \mathbf{Nom}_f$

4.3 Data Complexity of the IS- A_i s

In [5], we have shown how to express the DL *DL-Lite* for which QA (w.r.t. (T)CQs) is in **LogSpace** [8], with the CLs Lite-English. We want now to know which fragments

Table 4. Defining the $\{IS-A_i\}_{i \in [0,7]}$ CLs. $IS-A_i$, for all $i > 0$, contains also the assertions of $IS-A_0$.

	Assertions α_i	Example(s)
$IS-A_0$	$A \sqsubseteq A_1 \sqcap \dots \sqcap A_n$	Every businessman is a cunning man is a cunning man.
$IS-A_1$	$A \sqsubseteq \forall P:A$	Every herbivorous eats only herbs eats only herbs.
$IS-A_2$	$A_1 \sqcap \dots \sqcap A_n \sqsubseteq \forall P:(A_1 \sqcap \dots \sqcap A_k)$	Every Italian man drinks only strong coffee.
$IS-A_3$	$\exists P:A \sqsubseteq A_1 \sqcap \dots \sqcap A_n$ $\exists P^-:A \sqsubseteq A_1 \sqcap \dots \sqcap A_n$ $A \sqsubseteq \exists P$	Anybody who murders some person is a heartless killer. Anybody who is loved by some person is a happy person. Every driver drives something.
$IS-A_4$	$A_1 \sqcap \dots \sqcap A_n \sqsubseteq A_1 \sqcap \dots \sqcap A_k$ $\exists P:(A_1 \sqcap \dots \sqcap A_n) \sqsubseteq A_1 \sqcap \dots \sqcap A_k$	Every cruel man is a bad man. Anybody who runs some bankrupt company is a bad businessman.
$IS-A_5$	$\forall P:A \sqsubseteq A_1 \sqcap \dots \sqcap A_n$	Anybody who values only money is a greedy person.
$IS-A_6$	$A \sqsubseteq A_1 \sqcup \dots \sqcup A_n$	Every mammal is male or is female.
$IS-A_7$	$\neg A \sqsubseteq A_1 \sqcap \dots \sqcap A_n$	Anybody who is not selfish is a reasonable person.

of ACE-OWL and DL-English are (i) *maximal* w.r.t. tractable data complexity (i.e., in **PTime**), and hence scale to data, and (ii) *minimal* w.r.t. intractable data complexity (i.e., **coNP**-hard), and hence do not scale.

Theorem 3. *The data complexity of QA for (T)CQs is (i) in **LogSpace** for $IS-A_0$, (ii) **PTime**-complete for $IS-A_2$, $IS-A_3$, and $IS-A_4$ and (iii) **coNP**-complete for $IS-A_5$, $IS-A_6$, and $IS-A_7$.*

Proof. The CL $IS-A_0$ is subsumed by the CL Lite-English, which as we have shown elsewhere [5] expresses the DL *DL-Lite* for which QA w.r.t. CQs is in **LogSpace** in data complexity ([8], Theorem 2).

The lower bounds for $IS-A_2$, $IS-A_3$, and $IS-A_4$ follow from the results in [8]. For $IS-A_2$ the result is derived from Theorem 7, case 2. For $IS-A_3$, it is derived from Theorem 6, case 1. Finally, the lower bound for $IS-A_4$ follows from Theorem 7, case 3. Basically, this is because our CLs subsume the DLs for which those theorems hold. **PTime**-hardness in all three cases holds already for atomic queries. The complexity upper bounds, on the other hand, follow from results in [18] for the DL \mathcal{EL} , which subsumes the DL assertions $IS-A_2$, $IS-A_3$ and $IS-A_4$.

The lower bounds for $IS-A_5$, $IS-A_6$, and $IS-A_7$ follow also from [8]: for $IS-A_5$, we apply Theorem 8, case 3; for $IS-A_6$, we apply Theorem 8, case 2; and for $IS-A_7$, Theorem 8, case 1. In these three cases, TCQs are used to define a reduction from the **NP**-complete satisfiability problem for 2+2-clauses (defined any studied by [24]). The **coNP** upper bounds for these fragments, on the other hand, derive from the **coNP**

data complexity upper bounds for QA over expressive DLs (containing \mathcal{ALCT}) shown in [21] and hold, again, for CQs. \square

Theorem 4 (Data complexity of IS-A₁). *QA for IS-A₁ and (T)CQs is NLogSpace-complete w.r.t. data complexity.*

Proof. (Hardness): Calvanese et al. show in [8] (by reduction from the reachability problem for directed graphs) that any DL capable of expressing assertions of the form $A \sqsubseteq \forall P.A'$, or, equivalently, of the form $\exists P^-.A \sqsubseteq A'$, is **NLogSpace**-hard for QA. This result holds already for atomic queries. Note that such assertions are expressed in our fragments by sentences of the form “Every A Ps only A' s”, rather than by sentences like “Every A Ps every A' ”.

(Membership): Let $q(\bar{x}) \leftarrow \Phi(\bar{x}, \bar{y})$ be a fixed CQ, \bar{c} a fixed tuple, \mathcal{O} a fixed set of universally quantified IS-A₁ MRs and \mathcal{D} a set of facts s.t. $\#(\mathcal{D}) = n$. We will reduce QA for IS-A₁ to QA for linear Datalog, which is known to be in **NLogSpace** in data complexity [1]. The only inclusion assertions expressible in our fragment are $A \sqsubseteq B$ and $\exists R.A \sqsubseteq B$, which can be transformed into an (equivalent) set $\mathcal{P}_{\mathcal{O}}$ of clauses $\neg A(x) \vee B(x)$ and $\neg R(x, y) \vee \neg A(x) \vee B(y)$, called a linear Datalog *program*. On the other hand, CQ q might not be a linear Datalog goal, but its body $\Phi(\bar{x}, \bar{y})$ [1] consists of a conjunction of m atoms $A_1(\bar{z}_1) \wedge \dots \wedge A_m(\bar{z}_m)$, where $\bar{x} \cup \bar{y} = \bar{z}_1 \cup \dots \cup \bar{z}_m$. Note that, since q is fixed, m is constant (a fixed positive integer), and so are $|\bar{y}|$ and $|\bar{z}_i|$, for $i \in [1, m]$. If we were to transform such atoms into a family of atomic queries (which are linear Datalog goals), by means of some satisfaction-preserving reduction that requires only $\mathbf{O}(\log n)$ space, the data complexity upper bound would immediately follow.

Start by computing the program $\mathcal{P}_{\mathcal{O}}$ as described above. Since \mathcal{O} is fixed, transforming it into $\mathcal{P}_{\mathcal{O}}$ does not affect data complexity. We transform now $\Phi(\bar{x}, \bar{y})$, in space logarithmic in n , into a family of linear Datalog goals, thus reducing answering q over \mathcal{O} and \mathcal{D} to answering a family of atomic goals over $\mathcal{P}_{\mathcal{O}}$ and \mathcal{D} . Let $[\bar{c}/\bar{z}]$ denote the grounding mapping \bar{z} to the constants \bar{c} . Ground q by $\sigma := [\bar{c}/\bar{x}]$. Grounding q by σ , which returns the CQ of body $\Psi(\bar{c}, \bar{y})$, does not affect, once again, data complexity. Next, consider all the possible groundings $[\bar{c}'/\bar{y}]$ with $\bar{c}' \in \text{atom}(\mathcal{D})^{|\bar{y}|}$ and apply them to $\Psi(\bar{c}, \bar{y})$. There are $\mathbf{O}(n^{|\bar{y}|})$ such groundings. This yields a family of CQs of body $\Psi(\bar{c}, \bar{c}')$, whose atoms can be stored in a registry of $\mathbf{O}(\log n)$ size (we can encode such grounded atoms using $\mathbf{O}(\log n)$ bits). This reduction is sound and complete. Indeed

$$\langle \mathcal{O}, \mathcal{D} \rangle \models \Psi(\bar{c}, \bar{y}) \text{ iff } \mathcal{P}_{\mathcal{O}} \cup \mathcal{D} \models A_i(\bar{c}''), \text{ for all } i \in [1, k] \text{ and} \quad (\dagger) \\ \text{some } \bar{c}'' \in \text{atom}(\mathcal{D})^{|\bar{z}_i|} \text{ “compatible” with } \bar{c},$$

where by “compatible” we mean that \bar{c}'' coincides with \bar{c} on the distinguished variables (note that \bar{z}_i may contain *both* distinguished and non-distinguished variables).

The “ \Rightarrow ” direction is immediate. To prove the “ \Leftarrow ” direction, we reason as follows. Assume for contradiction that there exists an interpretation \mathcal{I} s.t. $\mathcal{I} \models \mathcal{P}_{\mathcal{O}} \cup \mathcal{D}$ but $\mathcal{I} \not\models A_i(\bar{c}'')$, for some $i \in [1, k]$ and every $\bar{c}'' \in \text{atom}(\mathcal{D})^{|\bar{z}_i|}$ “compatible”, again, with \bar{c} . Since $\mathcal{I} \models \mathcal{P}_{\mathcal{O}} \cup \mathcal{D}$, we have that $\mathcal{I} \models \langle \mathcal{O}, \mathcal{D} \rangle$ and $\mathcal{I} \models \Phi(\bar{c}, \bar{y})$. Therefore, for some grounding θ from \bar{y} into $\text{atom}(\mathcal{D})$, $\mathcal{I} \models A_i(\bar{z}_i)\sigma\theta$. Now, clearly, $\bar{c}'' = \bar{c}_1'' \cup \bar{c}_2''$ with $\bar{c}_1'' \subseteq \bar{c}$, and $\bar{z}_i = \bar{z}_{i1} \cup \bar{z}_{i2}$ with $\bar{z}_{i2} \subseteq \bar{x}$. Therefore, $\theta(\bar{z}_{i2}) \in \text{atom}(\mathcal{D})^{|\bar{z}_{i2}|}$. On the other hand, $\mathcal{I} \not\models A_i(\bar{c}'')$, for all \bar{c}'' . Hence, $\theta(\bar{z}_{i2}) \neq \bar{c}_2''$ and $\theta(\bar{z}_{i2}) \notin \text{atom}(\mathcal{D})^{|\bar{z}_{i2}|}$. Contradiction.

Table 5. Summary of data complexity results. For a comparison, note that ACE-OWL is **coNP**-hard (upper bounds for OWL-DL are not, as yet, known) [21].

CL	Data Complexity ((T)CQs)	CL	Data Complexity ((T)CQs)
IS-A ₀	LogSpace	EL-English	PTime -complete
IS-A ₁	NLogSpace -complete	IS-A ₅	coNP -complete
IS-A ₂	PTime -complete	IS-A ₆	coNP -complete
IS-A ₃	PTime -complete	IS-A ₇	coNP -complete
IS-A ₄	PTime -complete	DL-English	coNP -complete

The algorithm then proceeds by looping $\mathbf{O}(n^{|\bar{y}|})$ times over the $A(c'')$ s (stored in the $\mathbf{O}(\log n)$ registry), checking each time whether, for all $i \in [1, k]$, there exists some “compatible” \bar{c}'' s.t. $\mathcal{P}_O \cup \mathcal{D} \models A_i(\bar{c}'')$ holds. For each atom the algorithm runs a linear Datalog non-deterministic check that uses at most $\mathbf{O}(\log n)$ space. Clearly, such a non-deterministic algorithm decides QA using, overall, at most $\mathbf{O}(\log n)$ space. \square

We can now individuate the constructs of DL-English, and a fortiori of any CL expressing a **coNP**-hard ontology language that negatively affect the scalability of CL interfaces to ODBASs, namely:

- “only” in subject position (**coNP**-hardness of IS-A₅),
- disjunction in predicate position (**coNP**-hardness of IS-A₆),
- negation in subject position (**coNP**-hardness of IS-A₇).

4.4 EL-English: A Maximal Tractable Fragment of DL-English

The constructs from Fig 3 also allow us to identify *maximal* CLs contained in DL-English (and a fortiori ACE-OWL) w.r.t. scalability (i.e., tractable data complexity). By merging the (tractable) fragments IS-A_{*i*}, for $i \leq 4$, we essentially express the \mathcal{ELI} ontology language, with syntax (notice that the assertion $A \sqsubseteq \forall P : A'$ is equivalent to $\exists P^- : A \sqsubseteq A'$):

$$R \rightarrow P \mid P^- \qquad C \rightarrow \top \mid A \mid \exists R : C \mid C \sqcap C$$

That is, the DL where negation- and disjunction free existential concepts are allowed to arbitrarily nest on *both* sides of \sqsubseteq . QA for \mathcal{ELI} is **PTime**-complete in data complexity for (T)CQs [18] and thus induces a **PTime**-complete fragment of DL-English, that we term EL-English, which captures most of the constraints and axioms of real-world large-scale biomedical ontologies such as GALEN or SNOWMED [18]. EL-English is defined top-down by removing from G_{DL} the grammar rules for negation, disjunction, and universal quantification, and the negative function words. Whence:

Proposition 1. *QA for EL-English and (T)CQs is **PTime**-complete in data complexity.*

In such a CL arbitrary sentence subordination (and relatives), in combination with, existential quantification and conjunction among **VPs** or **Noms** is allowed. Universal quantification is highly controlled and negation and disjunction are ruled out.

4.5 Discussion

In general, intractability (w.r.t. data complexity) will arise in every CL that induces a partitioning of the data of the OBDAS's DB. This will happen whenever their MRs can simulate full negation, conjunction and disjunction: the technical **coNP**-hardness results from [8], on which we ground our own work, are based on this intuition. We can thus say that CLs like IS-A₅, IS-A₆, IS-A₇, and a fortiori DL-English (which contains them all) are “booleanly closed”. EL-English, on the other hand, by being a negation-free fragment of DL-English, remains tractable.

5 Conclusions

In this paper we have studied the data complexity of querying OBDASs in controlled English. We have shown that optimal (i.e., **LogSpace**) and tractable (i.e., **PTime**) data complexity (w.r.t. (T)CQs) is basically attained by constraining the behaviour of certain function words, namely those expressing universal restrictions, negations and disjunctions of concepts and roles in the underlying DL ontologies. They must not be allowed to occur *both* in the subject and predicate of CL sentences. Moreover, they can only occur in carefully sought positions of CL sentences. This is the case with the CLs IS-A₀, IS-A₁, IS-A₂, IS-A₃, IS-A₄, and EL-English. Relaxing the constraints on negation, disjunction, universal determiners, and pronouns, as in IS-A₅, IS-A₆ and IS-A₇, until the components in sentence subjects and predicates become symmetrical, as in DL-English (or more expressive CLs), yields **coNP**-hardness. Table 6 summarizes the computational properties associated to each such combination of (controlled) English function words.

Computational complexity will be higher if we consider *combined complexity* i.e., when we consider as inputs not only the data but also the ontology (and possibly the query). Satisfiability for *ALC_T* is **ExpTime**-complete (**ExpTime** is the class of problems solvable by a deterministic algorithm in exponential time) [3]. Since DL-English expresses *ALC_T* and for this DL QA of TCQs (but not CQs) reduces to unsatisfiability, this entails that QA for TCQs is **ExpTime**-complete. Instead, QA for CQs is **2ExpTime**-complete [9,17] in *ALC_T*, and hence in DL-English.

As related work we must mention the computational complexity results regarding the satisfiability problem for several fragments of (and CLs obtained from) English by Slavkovic [29] and Pratt and Third [23], from which upper and lower complexity bounds for the combined complexity of QA can be derived in some cases. Such fragments and CLs are orthogonal in expressiveness to the CLs studied in this paper. In the case of Pratt and Third's declarative fragments, termed fragments of English (FOEs),

Table 6. Data complexity of CL constructs

Tractable (PTime or less)	negation in predicate VPs , relatives in predicate VPs , conjunction in predicate VPs	relatives and conjunction in subject NPs and predicate VPs , but no negation
Intractable (coNP -hard)	relatives and negation in subject NPs and in predicate VPs	negation in subject NPs “only” in subject NPs

we have provided in [32] *data complexity* results for QA (albeit w.r.t. TCQs only). In particular, we show that “booleanly closed” FOEs are intractable not only w.r.t. combined complexity and/or satisfiability, but exhibit intractable data complexity for QA.

Acknowledgements. We thank Raffaella Bernardi and Norbert Fuchs for discussions and criticism regarding earlier versions of this paper.

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley, Reading (1995)
2. Androutsopoulos, I., Ritchie, G.D., Thanisch, P.: Natural language interfaces to databases - An introduction. *Journal of Natural Language Engineering* 1(1), 29–81 (1995)
3. Baader, F., Calvanese, D., Nardi, D., Patel-Schneider, P., McGuinness, D.: The Description Logic Handbook. Cambridge University Press, Cambridge (2003)
4. Barwise, J., Cooper, R.: Generalized quantifiers and natural language. *Linguistics and Philosophy* 4(2), 159–219 (1980)
5. Bernardi, R., Calvanese, D., Thorne, C.: Lite natural language. In: Proc. of the 7th Int. Workshop on Computational Semantics, IWCS-7 (2007)
6. Bernstein, A., Kaufman, E., Göring, A., Kiefer, C.: Querying ontologies: A controlled English interface for end-users. In: Proc. of the 4th Int. Semantic Web Conf. ISWC 2005 (2005)
7. Blum, A.: Microsoft English Query 7.5. automatic extraction of semantics from relational databases and OLAP cubes. In: Proc. of the 25th Int. Conf. on Very Large Databases, VLDB 1999 (1999)
8. Calvanese, D., de Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Data complexity of query answering in description logics. In: Proc. of the 10th Int. Conf. on the Principles of Knowledge Representation and Reasoning, KR 2006 (2006)
9. Calvanese, D., De Giacomo, G., Lenzerini, M.: On the decidability of query containment under constraints. In: Proc. of the 17th ACM Symposium on Principles of Database Systems (PODS 1998), pp. 149–158 (1998)
10. Carpenter, B.: Type-Logical Semantics. The MIT Press, Cambridge (1997)
11. Fuchs, N.E., Kaljurand, K.: Mapping Attempto Controlled English to OWL-DL. In: Demos and Posters of the 3rd European Semantic Web Conf. (ESWC 2006) (2006)
12. Fuchs, N.E., Kaljurand, K., Schneider, G.: Attempto Controlled English meets the challenges of knowledge representation, reasoning, interoperability and user interfaces. In: Proc. of the 19th Int. Florida Artificial Intelligence Research Society Conf. FLAIRS 2006 (2005), <http://www.ifi.unizh.ch/attempto/publications/>
13. Horrocks, I., Patel-Schneider, P.F., van Harmelen, F.: From *SHIQ* and RDF to OWL: The making of a web ontology language. *Journal on Web Semantics* 1(1), 7–26 (2003)
14. Jurafsky, D., Martin, J.: Speech and Language Processing. Prentice-Hall, Englewood Cliffs (2000)
15. Kaljurand, K.: Attempto Controlled English as a Semantic Web Language. PhD thesis, University of Tartu (2007), <http://attempto.ifi.uzh.ch/site/pubs/>
16. Kaljurand, K., Fuchs, N.E.: Verbalizing OWL in Attempto Controlled English. In: Proc. of 3rd OWL Experiences and Directions Workshop, OWLED 2007 (2007)
17. Lutz, C.: The complexity of conjunctive query answering in expressive description logics. In: Armando, A., Baumgartner, P., Dowek, G. (eds.) IJCAR 2008. LNCS (LNAI), vol. 5195, pp. 179–193. Springer, Heidelberg (2008)

18. Lutz, C., Krisnadhi, A.: Data complexity in the \mathcal{EL} family of DLs. In: Proc. of the 20th Int. Workshop on Description Logics, DL 2007 (2007)
19. Mador-Haim, S., Winter, Y., Braun, A.: Controlled language for geographical information system queries. In: Proc. of Inference in Computational Semantics 2006 (2006)
20. Montague, R.: Universal grammar. *Theoria* 36(3), 373–398 (1970)
21. Ortiz, M., Calvanese, D., Eiter, T.: Data complexity of query answering in expressive description logics. *Journal of Automated Reasoning* 41(1), 61–98 (2008)
22. Papadimitrou, C.: *Computational Complexity*. Addison Wesley - Longman, Amsterdam (1994)
23. Pratt, I., Third, A.: More fragments of language. *Notre Dame Journal of Formal Logic* 47(2), 151–177 (2006)
24. Schaerf, A.: On the complexity of the instance checking problem in concept languages with existential quantification. *Journal of Intelligent Information Systems* 2(3), 265–278 (1993)
25. Schwitter, R.: Creating and querying linguistically motivated ontologies. In: Proc. of the 2008 Knowledge Representation Workshop (KROW 2008) (2008)
26. Schwitter, R., Kaljurand, K., Cregan, A., Dolbear, C., Hart, G.: A comparison of three controlled natural languages for OWL 1.1. In: Proc. of the 4th Int. Workshop on OWL: Experiences and Directions (OWLED 2008), Washington (April 2008)
27. Schwitter, R., Ljungberg, A., Hood, D.: ECOLE - A look-ahead editor for a controlled language. In: Proc. of the 8th Int. Workshop of the European Association for Machine Translation and the 4th Controlled Language Applications Workshop (EAMT/CLAW 2003) (2003)
28. Schwitter, R., Tilbrook, M.: Let's talk in description logic via controlled language. In: Proc. of the 3rd Int. Workshop on Logic and Engineering of Natural Language Semantics (LENLS 2006) (2006)
29. Slavkovik, M.: Deep analysis for an interactive question answering system. Master's thesis, Free University of Bozen-Bolzano (2007)
30. Sowa, J.: *Knowledge Representation: Logical, Philosophical and Computational Foundations*. Brooks Cole Publishing Co., Pacific Grove (1999)
31. Sowa, J.: *Common Logic Controlled English*, draft (2004), <http://www.jfsowa.com/clce/specs.htm>
32. Thorne, C., Calvanese, D.: The data complexity of the syllogistic fragments of english. In: Proc. of the 2009 Amsterdam Colloquium, AC 2009 (2009)
33. Vardi, M.: The complexity of relational query languages. In: Proc. of the 14th Annual ACM Symposium on Theory of Computing, STOC 1982 (1982)

SBVR's Approach to Controlled Natural Language

Silvie Spreeuwenberg¹ and Keri Anderson Healy²

¹ LibRT, Amsterdam, Netherlands
silvie@librt.com

² BRCommunity.com, USA
kandersonhealy@brcommunity.com

Abstract. The “Semantics of Business Vocabulary and Business Rules” (SBVR 1.0) is one of the initial specifications in the OMG’s family of business-focused specifications. SBVR covers two aspects: Vocabulary (natural language ontology) and Rules (elements of guidance that govern actions). However, SBVR does not standardize any particular language for expressing vocabularies and rules. Instead, SBVR uses ‘semantic formulation’, which is a way of describing the semantic structure of statements and definitions. This approach of specifying structures of meaning, with its sound theoretical foundation of formal logic, provides a formal, language-independent means for capturing the semantics of a community’s body of shared meanings. By taking this approach, SBVR can support multiple forms of representation.

Keywords: Business Rules, Business Vocabulary, Semantics, Semantic Formulation, SBVR.

1 Introduction

Organizations are expected to deliver consistent behavior and communication to their customers in a fast and efficient way. To achieve this, organizations typically have implemented internal processes that guide the organization in answering a customer request. The process can often be started in multiple ways. In business terminology this is named ‘multi channel management’. Customers may make a request by visiting the organization, by making a phone call, by writing a letter, or by filling in a form on a website (multiple channels). One can imagine different people being involved in handling these requests, eventually in different places.

When the request is received, it must be processed in some way. Let’s take the example of a citizen requesting a taxi permit. The citizen may have been informed in a brochure about the conditions to get a taxi permit. He may ask for more information in a phone call, interacting with a representative working in a call-center. After this he may decide to submit a taxi permit request using a web form. An automated software service may now check if the provided information makes the applicant eligible for a taxi permit. The taxi permit is delivered to the applicant and an inspection is now scheduled in order to check if the taxi driver actually meets the taxi permit conditions in his operations.

In this taxi permit request example there are different people (or supporting software systems) that need to know the conditions to be met by the taxi driver for him to get a permit. The conditions in this example probably have a source in some legal document. In our example, the organization might have a hard time making sure everyone has the same understanding of how the text in the legal document is to be applied to the taxi permit applicant. One reason for this is that the different stakeholders reading the legal document might make different interpretations because text in natural language is often ambiguous.

The idea behind the introduction of a business rules management process in an organization is to remove such freedom of interpretation by delivering clear and unambiguous guidelines for both people and software systems. The legal text is interpreted by business analysts along with domain experts, and together they deliver clear and unambiguous business rules. Since the business rules are human readable but based on a formal vocabulary they can also help bridge the ‘knowledge gap’ between business and IT, resulting in decision support systems that are easier to build and maintain.

Specialists working in this field to support organizations in creating clear and unambiguous rules have been in need of a standard way to capture the meaning of guidance statements. This need was the driving force behind the development of a new standard, the “semantics for business vocabulary and rules” (SBVR), which is further described in this paper.

1.1 The Business Rules Approach

SBVR was conceived to provide a human-friendly approach to business rules. A core idea of the business rules approach is that rules build on facts, and facts build on concepts, expressed as terms. Terms express business concepts; facts make assertions about these concepts; rules constrain and support these facts.

This fundamental building-block perspective means that business rules rely on the definition of concepts and supporting vocabularies, and all of these should be in human-understandable terms. For example, to understand that an uncle is the brother of a parent, the business person should not have to parse a definitional rule expressed as:

hasParent(?x1,?x2) hasBrother(?x2,?x3) hasUncle(?x1,?x3)

However, not all “natural language” qualifies as a useful expression form in the business rules approach. The business rules approach does not encourage the statement of rules in the following *Legaldegook* form:

No savings and loan holding company, directly or indirectly or through one or more transactions, shall acquire control of an uninsured institution or retain, for more than one year after other than an insured institution or holding company thereof, the date any insured institution subsidiary becomes uninsured, control of such institution.

Business rules should be expressed in such a way that they can be validated for correctness by business people. Business rules should be expressed in such a way that they can be verified against each other for consistency. So to support this objective of

validation and verification, SBVR was also designed to provide a formal foundation for business rules.

1.2 SBVR

In January 2008, the Object Management Group (OMG) published “Semantics of Business Vocabulary and Business Rules” (SBVR 1.0), one of the initial specifications in the OMG's family of business-focused specifications. ‘Business’ in this context means ‘human’ — as contrasted with computer systems or technology. The term ‘business’ should be interpreted in its broadest sense, pertaining to any human/organizational activity, and not be interpreted narrowly (i.e., as only pertaining to commercial activity).

SBVR covers both aspects of the business rules approach:

- **Rules:** elements of guidance (policies, rules, advices) that govern business actions of an organization.
- **Vocabulary:** natural language ontology (‘terminology’ to ISO) — concepts and their representations (terms, names, definitions) as a cohesive set, rather than a simple list of terms and definitions.

However, SBVR does not standardize any *particular* language for expressing rules and vocabulary. Indeed, it is specifically *not* the intention of SBVR to mandate that any particular notation or grammar be used.

So if SBVR does not specify a language for stating rules and writing vocabularies, if it does not propose a set of diagramming conventions, and if it is not laying out a particular approach or methodology, what does SBVR standardize? First and foremost, SBVR specifies a metamodel for developing semantic models of business vocabularies and business rules. It establishes a vocabulary for talking about meaning, vocabularies, and business rules.

This metamodel and common vocabulary is the result of the collaborative efforts of twenty organizations, from seven countries across Europe, Asia, and North America. SBVR reflects a unique synthesis of four disciplines:

- **Terminology & Vocabulary:** drawing extensively from the ISO Terminology and Vocabulary work.
- **Linguistics & Communication:** tapping into Unisys' work on the linguistic expression of business rules.
- **Formal Logics:** incorporating foundational concepts from respected works on formal logics and mathematics, such as Object Role Modeling (ORM) and Common Logic (CL).
- **Pragmatic Business Practice:** reflecting the real-world business know-how of business practitioners.

SBVR supports the development and specification of business rules and vocabularies that reflect a business (rather than IT) perspective. These are rules and vocabularies that use the language of the business, not the terminology of IT models. Importantly, this is independent of whether or not the rules and vocabulary will be automated.

2 SBVR's Approach to Meaning

At the heart of SBVR's approach is its being concept-centric, rather than word-centric. To this is added the notion of 'semantic formulation', which is SBVR's way of describing the semantic structure of statements and definitions. It is important to note that semantic formulations are not *expressions* of meaning — rather they are *structures* of meaning (the logical composition of meaning). With its sound theoretical foundation of formal logic, SBVR provides a formal, language-independent syntax for capturing the semantics of a community's body of shared meanings.

These structures of meaning are not used directly by people. Instead, people will use a language that has a mapping to the structures of meaning in SBVR. This language can be in a graphical representation, but is most likely in a textual form. RuleSpeak® [1] and SBVR Structured English [11] are two example languages with a mapping to SBVR structures of meaning. They are controlled languages since they deal with a restricted subset of a language with respect to the mapping to the SBVR structures of meaning. However, the 'controlled' language can also be extended using the SBVR structures of meaning.

2.1 Core Notions in the Structuring of Meaning

The SBVR standard itself is described as an SBVR vocabulary and, as such, provides a start for describing a controlled language. The structures of meaning can be used to extend this controlled vocabulary with domain-specific terminology. In particular, the domain-specific concepts must be defined using the structures of meaning provided by SBVR. Core notions in the structuring of meaning are (among others) the following:

- *Noun concept*, defined as: concept that is the meaning of a noun or noun phrase
- *Individual concept*, defined as: concept that corresponds to only one thing
- *Verb concept*, defined as: concept that is the meaning of a verb phrase that involves one or more noun concepts and whose instances are all actualities

In a car rental business, typical noun concepts might be represented by the terms 'driver', 'vehicle', 'rental', etc. An example of individual concepts — usually only a small part of the total vocabulary — are 'Dollar' and 'Euro', each the name of a currency.

Verb concepts provide the ability to define connections between concepts that are of interest to the organization. These connections provide the business-level semantic structure required to find information about such relationships in text documents and relational databases, as well as providing the ability to specify business rules formally and unambiguously. For example, in a financial business, the connection between the concepts 'driver' and 'rental' might be defined by an associative fact type 'authorizes' ('rental authorizes driver'). Different kinds of verb concepts provide a powerful means to build ontologies that are semantically equivalent to those developed in Web Ontology Language (OWL). Drawing from [7], SBVR provides three kinds of hierarchical relations, to describe: *assortments* (relationship between individual and

general concept), *specializations* (hierarchical relationship between a concept and a category such that an instance of the concept is also an instance of the category), and *parts* (a given part being in the composition of a given whole).

By taking this approach, SBVR can support multiple forms of representation. For example, a fact type expressed in English can be readily understood in both its forward (person *rents* vehicle) and reverse (vehicle *is rented by* person) readings as being for the same meaning. Also, both noun and verb fact type forms can be interpreted as one meaning as in the following two example fact type expressions: 'person *has* phone number', 'phone number *of* person'.

2.2 Structuring of Meaning for Expressing Rules

An element of guidance is defined in SBVR as “means that guides, defines, or constrains some aspect of an enterprise.” This concept is then further analyzed, resulting in two ways to give guidance.

One is the *common sense* understanding of a ‘rule’. A rule guides behavior by removing some degree of freedom. The degree of freedom removed by a rule might concern the behavior of people (in the case of a behavioral business rule), or their understanding of concepts (in the case of a definitional rule).

An example of a behavioral business rule is the meaning of the sentence “A taxi driver must always take the shortest route to a destination.” This sentence clearly indicates that taking a route other than the shortest route is not allowed. Depending on the enforcement level of the rule the driver may decide to violate the rule, for example when the shortest route may not be the fastest route. So people can still potentially violate or ignore the rule — that is a matter of free will.

In a definitional rule, the restricting of freedom is built-in (i.e., “by definition”). An example is the meaning of the sentence “An official taxi driver always has a taxi permit.” By definition there is no official taxi driver without a taxi permit.

This *common sense* understanding of ‘rule’ should be contrasted with that for ‘advice’, where a degree of freedom is never removed, even potentially. An advice always confirms or reminds that some degree of freedom does exist or is allowed. That degree of freedom might concern the behavior of people (in the case of a behavioral business rule), or their understanding of concepts (in the case of a definitional rule). For example, the meaning of “It is permitted that an order be paid by cash.” is that such behavior is allowed — that indeed, paying by cash is acceptable. In other words, there is (or should be) no rule to the contrary.

When expressing guidance it is important to clearly indicate for what concept(s) freedom is removed (or confirmed). A guidance statement therefore uses a quantifier to express variations like:

- each driver must be (universal quantification)
- at least one passenger must be (existential quantification)
- at least two drivers must be (at-least-*n* quantification)
- etc.

In addition to the notion of quantifiers, SBVR also defines the notion of logical operators (and, or, if) so that these can be used as building blocks that have formal meaning in expressing guidance statements.

But there are limits to the kinds of meaning that can be expressed in SBVR guidance statements. There is no support for directives like the sentence ‘No smoking in this room please.’ A sentence in past (or future) tense, such as ‘... if a person was ill’, cannot be expressed (but we can say, ‘... if the person has been ill in the last year’). And the best practice to state a rule in one sentence results in the restriction that an attachment should be resolved within one sentence. We can refer to ‘the person’ if the concept ‘person’ has been introduced earlier in the sentence ... or if it is making reference to an individual concept, as in ‘the person John’.

2.3 Structures of Meaning That Improve Natural Readability

The need for human-readable guidance statements motivates the notions of objectification and nominalization in SBVR. An objectification uses a propositional expression to identify a state of affairs or event. In the sentence ‘A car assignment of a rental must occur before the pick-up date of the rental.’ the term ‘car assignment’ is an objectification of the fact type ‘car is assigned to rental’. By defining this objectification and the following fact types

- car assignment is a state of affairs
- state of affairs occurs before point in time
- state of affairs₁ occurs before state of affairs₂ occurs

we can express a rule of the form:

‘A car assignment of a rental must occur before the pick-up date of the rental.’

Without the objectification, a sentence to express this meaning would be very hard for an average human reader to follow:

“The point in time on which a car is assigned to a rental must occur before”

This example also illustrates the need for general notions for time. Such general notions are not part of SBVR, but it is expected that these kinds of vocabularies will become available in the future, such as the work described in [10]. These vocabularies may include ‘core ontologies’ that are developed in the semantic web community.

3 SBVR’s Approach to Representation

The approach of semantic formulation, with its logic grounding, supports two essential features of SBVR. First is the mapping of a semantic community’s body of shared meanings to the vocabularies (and thereby the expressions and communication forms) used by its speech communities. For example, a rule (such as one that prohibits crossing the railroad tracks) can be expressed in various national languages:

- Überschreiten der Gleise verboten
[in the German-speaking community]
- Défense de traverser les voies
[in the French-speaking community]
- Vietato attraversare i binari
[in the Italian-speaking community]

- Crossing the railway lines is prohibited
[in the English-speaking community]

A second feature is the mapping to XMI that enables interchange of concepts, facts, and business rules between languages (and supporting tools) that implement SBVR. (Additional detail on SBVR's use of XMI can be found in Clauses 13 and 15 of [11].) For example, below are equivalent expressions of the same rule, according to the language conventions of (respectively) RuleSpeak and SBVR Structured English.

- RuleSpeak: The renter of a vehicle must have exactly three phone numbers.
- SBVR Structured English: It is obligatory that the renter of a vehicle have exactly three phone numbers.

Each word or word phrase in the sentence is mapped to the equivalent element of meaning in SBVR. For the second sentence the following table provides that mapping.

<i>Representation in SBVR controlled English</i>	<i>SBVR element</i>
It is obligatory that	obligation formulation
the renter	noun concept
of	fact symbol in fact type 'renter of vehicle' ('vehicle has renter')
a	existential quantification
vehicle	noun concept
have	fact symbol in fact type 'renter has phone number'
exactly three	exactly-3 quantification
phone numbers	noun concept

To perform this mapping automatically, all words (or word phrases) have to be defined as SBVR elements of meaning. A parser that can deal with grammatical issues like plurals and tense and has an understanding of the language conventions is needed to create the mapping with the meaning. These components are and have been built by several vendors of supporting software tools.

The difference between RuleSpeak and SBVR Structured English is related to readability and ease of use. The design decisions one can make in creating a mapping from SBVR to a controlled language are worth more investigation, as are methods to evaluate the resulting language.

3.1 Representation of Guidance

Formulations of guidance in SBVR use the deontic and alethic modalities. Deontic modal operators are used for behavioral guidance (also known as 'operative' guidance); alethic modal operators are for definitional guidance (also known as 'structural' guidance). For each of these two categories there are equivalent patterns of expression

that may be used in a particular syntax to state the element of guidance. The following examples illustrate, using SBVR Structured English as the concrete syntax.

Consider the informally-stated behavioral rule, “Don’t rent a car without making a provisional charge against the renter’s credit card.” The table below provides this rule’s equivalent expression forms.

Modality	SBVR Structured English keywords	Guidance statement
prohibition	It is prohibited that... (or: ... must not ...)	It is prohibited that a rental is open if an estimated rental charge is not provisionally charged to the credit card of the renter of the rental.
obligation	It is obligatory that... (or: ... must ...)	It is obligatory that an estimated rental charge is provisionally charged to the credit card of the renter of an open rental.
restricted permission	It is permitted that ... only if ... (or: ... may ... only if)	It is permitted that a rental is open only if an estimated rental charge is provisionally charged to the credit card of the renter of the rental.

Furthermore, a restricted permission statement can always be restated as a conditional prohibition — for example, “It is not permitted that a rental is open if an estimated rental charge is not provisionally charged to the credit card of the renter of the rental.”

A similar case applies for definitional guidance. Consider the informally-stated definitional rule, “A rental is considered to be ‘open’ when the car has been picked up.” The table below provides this rule’s equivalent expression forms.

Modality	SBVR Structured English keywords	Guidance statement
necessity	It is necessary that... (or: ... always ...)	It is necessary that a rental is open is the rental car of the rental has been picked up.
impossibility	It is impossible that... (or: ... never not ...)	It is impossible that a rental is open if the rental car of the rental has not been picked up.
restricted possibility	It is possible that ... only if ... (or: ... can ... only if)	It is possible that a rental is open only if the rental car of the rental has been picked up.

Furthermore, a restricted possibility statement can always be restated as a conditional impossibility — for example, “It is not possible that a rental is open if the rental car of the rental has not been picked up.”

These cases of guidance illustrate ‘rules’ — restrictions on some degree of freedom or scope. SBVR also covers the case where guidance is an ‘advice’ — something that

confirms or reminds that some degree of freedom does exist or is allowed. And as with behavioral and definitional rules, expressions of advice have equivalent patterns of expression that are based on the deontic and alethic modalities.

Consider this informally-stated deontic advice, “A rental car doesn’t need to be returned to the place where it was picked up.” The table below provides equivalent expression forms for this element of advice.

<i>Modality</i>	<i>SBVR Structured English keywords</i>	<i>Guidance statement</i>
permission	It is permitted that... (or: ... may ...)	It is permitted that the drop-off branch of a rental is not the return branch of the rental.
non-obligation	It is not obligatory that... (or: ... need not ...)	It is not obligatory that the drop-off branch of a rental is the return branch of the rental.

Finally, consider this informally-stated alethic advice, “A rental can have multiple drivers.” The table below provides equivalent expression forms for this item of advice.

<i>Modality</i>	<i>SBVR Structured English keywords</i>	<i>Guidance statement</i>
possibility	It is possible that... (or: ... can ...)	It is possible that a rental has more than one driver.
non-necessity	It is not necessary that... (or: ... not always ...)	It is not necessary that a rental has exactly one driver.

3.2 SBVR Is Not a Controlled Language

SBVR is not itself a controlled language so it is up to each SBVR-implementing language or notation to specify its formal mechanisms. SBVR Structured English accomplishes this by a combination of styling and key words.

SBVR Structured English defines four formatting styles, as follows:

- terms: underlined teal text is used for designations of noun concepts that are defined in the vocabulary. A term typically begins with a lower-case letter and is defined in singular form — for example, renter, branch, currency. Plural forms are implicitly available.
- names: double-underlined green text is used to reference individual concepts that are defined in the vocabulary. Names tend to be proper nouns and thus typically begin with an upper-case letter. For example, Switzerland, Euro.
- verbs: blue italic text is used for designations of fact types — usually a verb, preposition, or combination thereof. Such a designation is defined in the context of a fact type form — for example, branch owns rental car.

- key words: orange text is used for linguistic symbols used to construct statements and definitions in combination with other designations. Some punctuation marks (single quote, period) also use key word styling when given special interpretation — for example, the use of single quotes around a fact type form or concept that is being mentioned.

Key words and phrases are defined for expressing various kinds of logical formation. (In the following selected examples, the letters ‘*n*’ and ‘*m*’ represent use of a literal whole number, and the letters ‘*p*’ and ‘*q*’ represent expressions of propositions.)

- quantification
 - each: universal quantification
 - some: existential quantification
 - at least *n*: at-least-*n* quantification
- logical operations
 - *p* and *q*: conjunction
 - *p* or *q*: disjunction
 - it is not the case that *p*: logical negation
 - if *p* then *q*: implication
- modal operations
 - it is obligatory that *p*: obligation formulation
 - ... must ...: obligation formulation
 - ... must not ...: obligation formulation embedding a logical negation
 - it is necessary that *p*: necessity formulation
- other key words
 - a, an: universal or existential quantification, depending on context based on rules of English.
 - the: used with a designation to make a pronominal reference to a previous use of the same designation (formally a binding to a variable of a quantification), for example,
 - a driver is qualified if the driver is licensed.
 - Also, used to introduce a name of an individual thing or of a definite description, for example,
 - the country ‘Switzerland’ has mountains.
 - that: used as a binding to a variable (as with ‘the’) when preceding a designation for a noun concept, for example,
 - a rental is open if the rental car of that rental has been picked up.
 - Also, used to introduce a restriction on things denoted by the previous designation based on facts about them (when after a designation for a noun concept and before a designation for a fact type), for example,
 - a car that is qualified.
 - Also, used to introduce nominalization of a proposition or objectification (when followed by a propositional statement), for example,
 - If a car is assigned to a rental then the rental report of the rental must specify that the car is assigned to the rental.

- *of*: used as a shorthand for *that is of*. It is implicitly assumed that the statement ‘person *has* age’ has the inverse reading ‘age *of* person’ (‘age *that is of* person’).

3.3 SBVR Supports Both Formal and Informal Expression

So far, the examples have presented formally-expressed statements of guidance, in the conventions of SBVR Structured English. However, the SBVR initiative is intended to capture business facts and business rules that may be expressed either formally or informally. Business rule expressions are classified as formal only if they are expressed purely in terms of concepts in the pre-declared schema for the business domain, using conventions of the notation as well as certain logical/mathematical operators, quantifiers, etc. Such formal statements of rules may then be transformed into logical formulations that are used for exchange with other rules-based software tools.

However, even informal statements of rules may be exchanged, as un-interpreted comments, and are useful in the capture and authoring of business rules and vocabularies. In developing business rules and vocabularies the objectives of both the formal (*formalist*) and informal (*naturalist*) approaches to controlled natural language are appealing. Each SBVR-implementing notation will have its bias toward one or the other. Not surprisingly the SBVR Structured English objectives have a strong overlap with the objectives of the formalist approach to CNL. SBVR Structured English acknowledges that it

“[uses] English that maps mechanically to SBVR concepts. It is not meant to offer all of the variety of common English, but rather, it uses a small number of English structures and common words to provide a simple and straightforward mapping.”

On the other hand, the objectives of RuleSpeak have more overlap with the *naturalist* approach to CNL. As RuleSpeak says about itself,

“The purpose of [RuleSpeak’s] Sentence Forms is to ensure that written Business Rules are more easily understood. They also help ensure that different practitioners working on a large set of Business Rules express the same ideas in the same way. Such consistency would not be possible if Business Rules were expressed in a completely ‘free-form’ manner.”

If the compilation of business rules and vocabularies is viewed as a creative process, the ability to support both the naturalist approach and the formalist approach is valuable. In the initial stage, the earliest versions penned by a business expert are often not the complete, truthful, non-ambiguous, and perfect verbalizations of rules and definitions. Often they take the form of a *braindump* of the most important conditions that apply to a situation. In this stage a naturalist approach to controlled language may help the user to structure the resulting expressions by offering support in the removal of ambiguous structures from the sentences and by suggesting vocabulary enhancements that ensure completeness.

Only when this first step of the authoring process is finished can CNL of the formalist kind be introduced. Often this results in updating the expressions to make

them machine-processable. It is also at this stage where verification checks can be performed, such as looking for conflicts and subsumptions. Validation can take place after completing both stages.

While it may be tempting for the experienced rule author with a background in formal logics to skip the first step and write the rules directly in the formalist approach this has its drawbacks. Omitting the first step blocks the creative process and can lead to rules that erroneously simplify the situation at hand.

4 Logic Grounding

SBVR's logic foundation is first-order predicate logic with some restricted extensions into higher-order logics, along with some limited extensions into modal logic — notably some deontic forms (for expressing obligation and prohibition) and alethic forms (for expressing necessities and possibilities).

Table 1 provides a table of SBVR concepts to illustrate their equivalent structure in terms of formal logic as defined Common Logic (CL) [8]. This table consists of a representative set of SBVR concepts — a more complete table can be found in Clause 10.2 of the SBVR standard.

The mapping between SBVR and formal logic is not completely described in the SBVR Version 1.0. A more complete description is expected in a future release of the document.

The items shown under the heading *SBVR Terms* are all “meanings” while those shown under the *ISO CL Term* heading are “representations of meanings.” Therefore there is a one-to-many relationship between SBVR Terms (as meanings) and CLTerms (as representations of meanings), i.e., there can be multiple CL representations of one SBVR meaning.

Table 1.

SBVR Term	ISO CL Term (or equivalent expression)
fact type	unary predicate defining the type for a functional term or atomic sentence
fact type (binary fact type)	unary predicate defining the type for a functional term or atomic sentence that has exactly two arguments
individual concept	name
object type	unary predicate
proposition	sentence with an interpretation
concept incorporates characteristic	sentence (forall (u)(implies(characteristic u)(concept u)))
concept1 specializes concept2	(noun concept) (forall (c1 c2) (if (specializes c1 c2) (forall (x) (if (c1 x) (c2 x)))))) (forall (c1 c2) (if (and (specializes c1 c2) (specializes c2 c3)) (specializes c1 c3)))
integer	atom (integer x)
existential quantification	quantified sentence of type existential

SBVR allows for higher-order logic, but users may want to restrict themselves to first-order logic by adopting specific conventions in the following three aspects of fact modeling: categorization schemes, un-normalized structures, and crossing levels/metalevels within the same model. Halpin describes in [5] some suggested ways to avoid higher-order types.

When higher-order constructs are used, SBVR suggests restricting their interpretation using Henkin semantics (e.g., for dealing with categorization types). Specifics on these higher-order logic restrictions are documented in Clause 10.1.1.9 of the SBVR standard.

4.1 Modal Logic

Narrowly construed, modal logic studies reasoning that involves the use of the expressions 'necessarily' and 'possibly'. However, the term 'modal logic' is used more broadly to cover a family of logics with similar rules and a variety of different symbols [13]. SBVR uses four modal operators (and their negations) to capture the meaning of behavioral business rules, definitional rules, and advises (see Table 2).

Table 2.

Modality	Model operator	Equivalence
<i>Deontic</i>	<i>Obligation</i> : O	$Op \equiv \sim P \sim p.$
<i>Alethic</i>	<i>Necessity</i> : \Box	$\Box p \equiv \sim \Diamond \sim p.$
<i>Deontic</i>	<i>Permission</i> : P	$Pp \equiv \sim O \sim p.$
<i>Alethic</i>	<i>Possibility</i> : \Diamond	$\Diamond p \equiv \sim \Box \sim p.$

SBVR's use of modal logics yields provably-equivalent patterns of rule expression. For example, a given business rule can be stated in the form of prohibition, obligation, or restricted permission and be assured to represent the same underlying meaning. Again, these are three semantically-equivalent natural language expressions of one rule:

- It is prohibited that an open rental has an intoxicated driver.
- It is obligatory that no open rental has an intoxicated driver.
- It is permitted that a rental be open only if the rental does not have an intoxicated driver.

Assuming that the characteristics 'person is intoxicated' and 'rental is open' are part of the vocabulary and that 'driver' specializes 'person', the semantic formulation underlying these statements can be expressed as:

It is obligatory that

. Not

... Exists v1 : 'rental' where 'rental is open'(v1)

... Exists v2 : 'driver' where 'person is intoxicated'(v2)

... 'rental has driver'(v1, v2)

Or, equivalently:

It is obligatory that

. For all $v1$: ‘rental’ where ‘rental is open’(v1)

.. For all $v2$: ‘driver’ where ‘person is intoxicated’(v2)

... Not

.... ‘rental has driver’(v1, v2)

4.2 SBVR and Reasoning

The SBVR standard does not standardize reasoning or inference based on the SBVR sentences. The standard does give some directions to consider when reasoning based on sentences described in SBVR.

One is that a conceptual model containing both a conceptual schema (fact types and rules) and a population of facts (*fact* that is of a particular *fact type* in the conceptual schema) cannot change. Any change to a conceptual model, including any change to any fact in the fact population, creates a different conceptual model. Each conceptual model is distinct and independent, although there may be relationships between conceptual models that share the same conceptual schema.

SBVR uses structures from logic that have been well studied. For some of the constructs used it is known that a sound and complete reasoner cannot be constructed. For example, there are known issues for reasoners dealing with higher-order logic in combination with open world semantics. In such cases one can restrict the use of SBVR to those aspects that are first order if that is what is desired.

We expect that multiple SBVR reasoners can be developed with slightly different usage scenarios. A monotonic reasoner may be applied to some aspect of SBVR (definitional rules) that assesses true statements. But in many areas, the business is not interested in what is true for all time. They are interested in what other facts we can infer (creating a new conceptual model) based on what is currently known (typical inference engine behavior). In some cases they may want to change currently true but unfavorable situations into future favorable situations. We have also seen cases where business people just want to list all rules that apply to a specific situation (check list generator).

5 Further Research

SBVR provides the foundation to define the meaning of a controlled natural language to write guidance statements. At this moment, different CNLs based on SBVR for different natural languages have been developed. For example the RuleSpeak sentence patterns have recently been translated to Dutch, German, and Spanish. During this translation work, a need to evaluate the resulting CNL for a particular user group was identified. Defining good metrics and methods to evaluate the learnability and usability of a CNL will be a valuable instrument to make SBVR successful in practical environments. We are happy that work in that direction has been presented in the CNL workshop.

A successful adoption of the SBVR standard is not possible without supporting tools that help people write sentences with a formal meaning in SBVR. These tools should also be able to process and analyze the SBVR statements to support day-to-day

decision making in organizations. Such systems may need a marriage between the traditional production systems (production rule systems) based on a Rete algorithm (or something similar), constraint propagators, and reasoning based on description logic. Such research needs an interdisciplinary approach, like the SBVR standard, between practitioners in logic, reasoning, language, and software engineering. We hope that this paper helps initiate such research efforts.

Acknowledgments

We would like to express our appreciation to all the members of the team who contributed to the SBVR specification. It is their willingness to share knowledge and their individual dedication to quality work that is making this body of work a success. In particular, we would like to thank those who have devoted their time and effort to reviewing, and improving, this paper.

References

1. Business Rule Solutions: BRS RuleSpeak® Practitioner's Kit. Business Rule Solutions, LLC (2001-2004), <http://www.rulespeak.com>
2. Clark, P., Harrison, P., Murray, W., Thompson, J.: Naturalness vs. Predictability: A Key Debate in Controlled Languages. In: Fuchs, N.E. (ed.) Pre-Proceedings Workshop on Controlled Natural Language, Marettimo Island (2009)
3. Girle, R.A.: Modal Logics and Philosophy. McGill-Queen's University Press (2000)
4. Halpin, T.A., Girle, R.A.: Deductive Logic, 2nd edn. Logiquest, Brisbane (1981)
5. Halpin, T.A.: Information Modeling and Higher-Order Types. In: Grundspenkis, J., Kirkova, M. (eds.) Proc. CAiSE 2004 Workshops, vol. 1, pp. 233–248. Riga Tech. University (2004), <http://www.orm.net/pdf/EMMSAD2004.pdf>
6. Halpin, T.A.: Object-Role Modeling (ORM/NIAM): An Overview. Springer, San Francisco (2000), <http://www.orm.net/pdf/springer.pdf>
7. International Organization for Standardization (ISO): Terminology work — Vocabulary - Part 1: Theory and Application. English/French ed., ISO (2000)
8. International Organization for Standardization (ISO): Information technology — Common Logic (CL) — A Framework for a Family of Logic-Based Languages, ISO (2005)
9. Larson, R., Segal, G.: Knowledge of Meaning: An Introduction to Semantic Theory. The MIT Press, Cambridge (1995)
10. Object Management Group (OMG): Date-Time Foundation Vocabulary, Request for Proposal. OMG document bmi/2008-03-02, http://www.omg.org/techprocess/meetings/schedule/Date-Time_Foundation_Vocabulary_RFP.html
11. Object Management Group (OMG): Semantics of Business Vocabulary and Business Rules (SBVR), v1.0. OMG (January 2008), <http://www.omg.org/spec/SBVR/1.0/PDF>
12. Nijssen, S., Bijlsma, R.: A Conceptual Structure of Knowledge as a Basis for Instructional Designs. In: Kinshuk, R., Koper, P., Kommers, P., Kirschner, D., Sampson, G., Dideren, W.E. (eds.) ICALT 2006, IEEE: 6th Int. Conf. on Advanced Learning Technologies, pp. 7–9. IEEE, Los Alamitos (2006)
13. Zalta, E.N. (ed.): Stanford Encyclopedia of Philosophy. The Metaphysics Research Lab, Center for the Study of Language and Information. Stanford University, <http://plato.stanford.edu/>

The Naproche Project

Controlled Natural Language Proof Checking of Mathematical Texts

Marcos Cramer, Bernhard Fisseni, Peter Koepke, Daniel Kühlwein,
Bernhard Schröder, and Jip Veldman

University of Bonn and University of Duisburg-Essen
{cramer, koepke, kuehlwei, veldman}@math.uni-bonn.de,
{bernhard.fisseni, bernhard.schroeder}@uni-due.de
<http://www.naproche.net>

Abstract. This paper discusses the semi-formal language of mathematics and presents the Naproche CNL, a controlled natural language for mathematical authoring. Proof Representation Structures, an adaptation of Discourse Representation Structures, are used to represent the semantics of texts written in the Naproche CNL. We discuss how the Naproche CNL can be used in formal mathematics, and present our prototypical Naproche system, a computer program for parsing texts in the Naproche CNL and checking the proofs in them for logical correctness.

Keywords: CNL, mathematical language, formal mathematics, proof checking.

1 Introduction

The Naproche project¹ (NATural language PROof CHEcking) studies the semi-formal language of mathematics (SFLM) as used in mathematical journals and textbooks from the perspectives of linguistics, logic and mathematics. A central goal of Naproche is to develop and implement a controlled natural language (CNL) for mathematical texts which can be transformed automatically into equivalent formulae of first-order logic using methods of computational linguistics.

One possible application of this mathematical CNL is to make formal mathematics more readable to the average mathematician; this application is fundamental to most other applications and therefore is currently the focus of Naproche development. In order to show the feasibility of this goal, we develop the prototypical *Naproche system*, which can automatically check texts written in the Naproche CNL for logical correctness. We test this system by reformulating parts of mathematical textbooks and the basics of mathematical theories in the Naproche CNL and automatically checking the resulting texts.

¹ Naproche is a joint initiative of PETER KOEPKE (Mathematics, University of Bonn) and BERNHARD SCHRÖDER (Linguistics, University of Duisburg-Essen). The Naproche system is technically supported by GREGOR BÜCHEL from the University of Applied Sciences in Cologne.

Once Naproche has sufficiently broad coverage, we also plan to use it as a tool that can help undergraduate students to learn how to write formally correct proofs and thus get used to (a subset of) SFML. This possible application of Naproche also makes it more evident why we are emphasising a lot on the naturalness of our CNL.

We first discuss the relevant features of SFLM and then discuss the Naproche CNL, Proof Representation Structures and the translation to first order logic. Finally, we describe how proof checking in Naproche can be of use to the field of formal mathematics and discuss further development of Naproche.

2 The Semi-Formal Language of Mathematics

As an example of the semi-formal language of mathematics (SFLM), we cite a proof for the theorem “ $\sqrt{2}$ is irrational” from HARDY-WRIGHT’s introduction to number theory [8].

If $\sqrt{2}$ is rational, then the equation $a^2 = 2b^2$ is soluble in integers a, b with $(a, b) = 1$. Hence a^2 is even, and therefore a is even. If $a = 2c$, then $4c^2 = 2b^2$, $2c^2 = b^2$, and b is also even, contrary to the hypothesis that $(a, b) = 1$.

SFLM incorporates the syntax and semantics of the general natural language, so that it takes over its complexity and some of its ambiguities. However, SFLM texts are distinguished from common language texts by several characteristics:

- They combine natural language expressions with mathematical symbols and formulae, which can syntactically function as noun phrases or sub-propositions.
- Constructions which are hard to disambiguate are generally avoided.
- Mathematical symbols can be used for disambiguation, e.g. by use of variables instead of anaphoric pronouns.
- Assumptions can be introduced and retracted. For example, the proof cited above is a proof by contradiction: At the beginning, it is assumed that $\sqrt{2}$ is rational. The claims that follow are understood to be relativised to this assumption. Finally the assumption leads to a contradiction, and is retracted to conclude that $\sqrt{2}$ is irrational.
- Mathematical texts are highly structured. At a global level, they are commonly divided into building blocks like definitions, lemmas, theorems and proofs. Inside a proof, assumptions can be nested into other assumptions, so that the scopes of assumptions define a hierarchical proof structure.
- Definitions add new symbols and expressions to the vocabulary and fix their meaning.
- Proof steps are commonly justified by referring to results in other texts, or previous passages in the same text. So there are intertextual and intratextual references.
- Furthermore, SFML texts often contain commentaries and hints which guide the reader through the process of the proof, e.g. by indicating the method of proof (“by contradiction”, “by induction”) or giving analogies.

3 The Naproche CNL

The Naproche CNL is currently a small but functional subset of SFLM which includes some of the above mentioned characteristics of SFLM. We first discuss the syntax of the Naproche CNL, and finally present a CNL version of the example proof from Section 2. A more detailed specification of the CNL syntax can be found in [5].

In the Naproche CNL we distinguish between *macrostructure* (general text structure) and *microstructure* (grammatical structure in a sentence).

3.1 Macrostructure

A Naproche text is structured by structure markers: *Axiom*, *Theorem*, *Lemma*, *Proof*, *Qed* and *Case*. For example, a theorem is presented after the structure marker *Theorem*, and its proof follows between the structure markers *Proof* and *Qed*. In a proof by case distinction, each case starts with the word *Case*, and the end of a case distinction can be marked by a sentence starting with *in both cases* or *in all cases*.

Assumptions are always introduced by an assumption trigger (e.g. *let*, *consider*, *assume that*). A sentence starting with *thus* always retracts the most recently introduced assumption that hasn't yet been retracted. When finishing a proof with a *qed*, all assumptions introduced in the proof get retracted.

Definitions can be used to define the meanings of constants, function symbols, relation symbols, verbs, adjectives and nouns. They can be marked by the words *definition* or *define*.

Here is an extract from our reformulation of EDMUND LANDAU's *Grundlagen der Analysis* [12], with all structure markers marked in bold font:

Axiom 3: For every x , $x' \neq 1$.

Axiom 4: If $x' = y'$, then $x = y$.

Theorem 1: If $x \neq y$ then $x' \neq y'$.

Proof: Assume that $x \neq y$ and $x' = y'$. Then by axiom 4, $x = y$. **Qed.**

Theorem 2: For all x $x' \neq x$.

Proof: By axiom 3, $1' \neq 1$. **Suppose** $x' \neq x$. Then by theorem 1, $(x')' \neq x'$. **Thus** by induction, for all x $x' \neq x$. **Qed.**

Definition 1: Define + recursively:

$x + 1 = x'$. $x + y' = (x + y)'$.

Additionally we present an example of a proof by case distinction in the Naproche CNL:

Theorem: No square number is prime.

Proof: Let n be a square number.

Case 1: $n = 0$ or $n = 1$. Then n is not prime.

Case 2: $n \neq 0$ and $n \neq 1$. Then there is an m such that $n = m^2$. Then $m \neq 1$ and $m \neq n$. m divides n , i.e. n is not prime.

So **in both cases** n is not prime. **Qed.**

3.2 Microstructure

Mathematical terms and formulae can be combined with natural language expressions to form statements, definitions and assumptions: The use of nouns, adjectives, verbs, natural language connectives (e.g. *and*, *or*, *i.e.*, *if*, *iff*) and natural language quantification (e.g. *for all*, *there is*, *every*, *some*, *no*) works as in natural English, only with some syntactical limitations similar to those in Attempto Controlled English [7]. Sentences in a proof can start with words like *then*, *hence*, *therefore* etc. Mathematical terms can function as noun phrases, and mathematical formulae can function as sentences or sub-clauses. The language of mathematical formulae and terms that can be used in the Naproche CNL is a first order language with function symbols, including some syntactic sugar that is common in the mathematical formulae and terms found in SFLM.

3.3 Coverage

The Naproche CNL in its present state does not cover all constructs of SFLM, but most texts can be rewritten in the Naproche CNL in such a way that they resemble the original text and still read like SFLM.

In our biggest test so far, we translated the first chapter of LANDAU's *Grundlagen der Analysis* [12] (17 pages) into the Naproche CNL. The resulting text can be seen online [4], and stays close to the original. Another example of a reformulation in the Naproche CNL can be seen in the following subsection.

We are continuously extending the language to allow more and more SFLM constructs.

3.4 $\sqrt{2}$ in Naproche CNL

The irrationality proof can be reformulated in the Naproche CNL as follows:

Theorem: $\sqrt{2}$ is irrational.

Proof:

Assume that $\sqrt{2}$ is rational. Then there are integers a, b such that $a^2 = 2 \cdot b^2$ and $\gcd(a, b) = 1$. Hence a^2 is even, and therefore a is even. So there is an integer c such that $a = 2 \cdot c$. Then $4 \cdot c^2 = 2 \cdot b^2$, $2 \cdot c^2 = b^2$, and b is even. Contradiction. Qed.

Naproche uses L^AT_EX input, so the actual input used would be the L^AT_EX code that was used to typeset the above text. We are also working on the alternative of using the mathematical WYSIWYG editor TeXmacs [15].

4 Proof Representation Structures

Texts written in the Naproche CNL are translated into *Proof Representation Structures* or PRSs (see [10], [11]). PRSs are Discourse Representation Structures (DRSs, [9]), which are enriched in such a way as to represent the distinguishing characteristics of SFLM discussed in Section 2.

A PRS has five constituents: An identification number, a list of discourse referents, a list of mathematical referents, a list of textual referents and an ordered list of conditions². Similar to DRSs, we can display PRSs as “boxes” (Figure 1).

i	
d_1, \dots, d_m	m_1, \dots, m_n
c_1	
\vdots	
c_l	
r_1, \dots, r_p	

Fig. 1. A PRS with identification number i , discourse referents d_1, \dots, d_m , mathematical referents m_1, \dots, m_n , conditions c_1, \dots, c_l and textual referents r_1, \dots, r_p

Mathematical referents are the terms and formulae which appear in the text. As in DRSs, discourse referents are used to identify objects in the domain of the discourse. However, the domain contains two kinds of objects: mathematical objects like numbers or sets, and the symbols and formulae which are used to refer to or make claims about mathematical objects. Discourse referents can identify objects of either kind.

PRSs have identification numbers, so that we can refer to them from other points in the discourse. The textual referents indicate the intratextual and intertextual references.

Just as in the case of DRSs, PRSs and PRS conditions are defined recursively: Let A, B, B_1, \dots, B_n be PRSs, d, d_1, \dots, d_n discourse referents and m a mathematical referent. Then

- for any n -ary predicate p (e.g. expressed by adjectives and noun phrases in predicative use and verbs in SFLM), $p(d_1, \dots, d_n)$ is a condition.
- $holds(d)$ is a condition representing the claim that the formula referenced by d is true.
- $math_id(d, m)$ is a condition which binds a discourse referent to a mathematical referent (a formula or a term).
- A is a condition.
- $\neg A$ is a condition, representing a negation.
- $A \Rightarrow B$ is a condition, representing an assumption (A) and the set of claims made inside the scope of this assumption (B).
- $A \Leftrightarrow B$ is a condition, representing a logical equivalence.
- $A \Leftarrow B$ is a condition, representing a reversed conditional (i.e. of the form “...if ...”)
- $A \vee B$ is a condition, representing an inclusive disjunction.
- $>< (A_1, \dots, A_n)$ is a condition, representing an exclusive disjunction, i.e. the claim that precisely one of A_1, \dots, A_n holds.

² The order of the conditions in a PRS reflects the argument structure of a proof and is relevant to the PRS semantics. This was in part inspired by ASHER’s SDRT [1].

- $<>$ (A_1, \dots, A_n) is a condition, representing the claim that at most one of A_1, \dots, A_n holds.
- $A := B$ is a condition, representing a definition of a predicate.
- $f : A \rightarrow B$ is a condition, representing a definition of a function or constant (where A fixes the scope of the function and B specifies its defining equality or property).
- *contradiction* is a condition, representing a contradiction.

4.1 PRS Construction

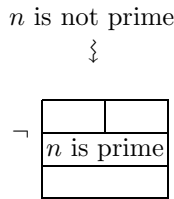
The algorithm creating PRSs from CNL proceeds sequentially [10]:

- It starts with the empty PRS.
- Structure markers open special *structure PRSs*.
- An assumption triggers a condition of the form $A \Rightarrow B$, where A contains the assumption and B contains the representation of all claims made inside the scope of that assumption.
- Representations of single sentences are produced in a way similar to a standard threading construction algorithm (see [2]).

We clarify both the PRS construction algorithm and the functions of the various kinds of PRS conditions by presenting examples which show how the most important PRS conditions are constructed from input text:

Negation conditions

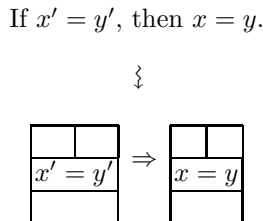
Negation conditions work just as in Discourse Representation Theory:



The downward arrow in this and the following examples roughly corresponds to one step of the PRS construction algorithm. The “ n is prime” in this PRS would still be processed further.

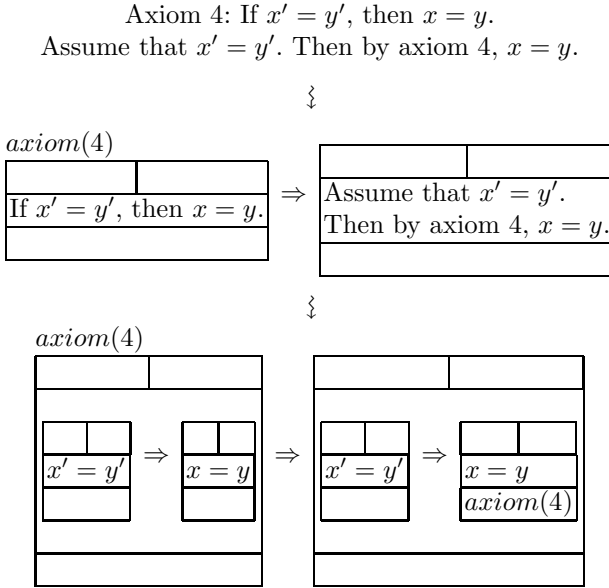
Implication conditions

PRS conditions of the form $A \Rightarrow B$ are called *implication conditions*. One function of them is to represent conditionals of the form *If ... then ...*:



When an assumption is introduced in a proof, and then something is derived from that assumption, this is also represented by an implication condition: The assumption is placed in the left PRS of the condition, and all proof steps until the assumption gets retracted are placed in the right PRS. Axioms are treated like assumptions that never get retracted.

The following example shows how different linguistic constructions can yield implication conditions, and how these can be nested inside one another:

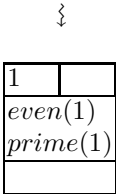


Note that the *axiom*(4) entry in the second PRS is not a condition, but a textual referent to a premise (see 5.3).

Predicate conditions

Predicate conditions work just like in DRT:

There is an even prime.



Note that we use natural numbers (1, 2, 3, ...) as discourse referents.

math_id conditions

As in DRT, all objects that are talked about are represented by discourse referents. Additionally, the mathematical terms (e.g. the variables) used in the text

are listed in the PRS as mathematical referents. It is therefore necessary to indicate which discourse referent refers to the same object as a certain mathematical referent. For this we use *math_id* conditions:

There are integers m, n such that m divides n .

⋈

1, 2	m, n
<i>integer</i> (1)	
<i>math_id</i> (1, m)	
<i>integer</i> (2)	
<i>math_id</i> (2, n)	
<i>divide</i> (1, 2)	

***holds* conditions**

When a mathematical formula is used, this gets represented by a *holds* condition, which indicated the truth of the formula:

Then $m \neq 1$ and $m \neq n$.

⋈

1, 2	$m \neq 1, m \neq n$
<i>math_id</i> (1, $m \neq 1$)	
<i>holds</i> (1)	
<i>math_id</i> (2, $m \neq n$)	
<i>holds</i> (2)	

Definition conditions

There are two special kinds of conditions for definitions. Here we concentrate on just one of these two kinds; conditions of this kind are of the form “ $A := B$ ” and represent definitions by biconditionals:

Define m to be square iff there is an integer n such that $m = n^2$.

⋈

1	m	:=	2, 3	$n, m = n^2$
$id(1, m)$			$id(2, n)$	
$square(1)$			$id(3, m = n^2)$	
			$holds(3)$	

Bare PRSs as conditions

Contrary to the case of DRSs, a bare PRS can be a direct condition of a PRS. This allows us to represent in a PRS the structure of a text divided into building blocks (definitions, lemmas, theorems, proofs) by structure markers.

Theorem 1: If $x \neq y$ then $x' \neq y'$.

Proof: Assume that $x \neq y$ and $x' = y'$. Then by axiom 4, $x = y$. Qed.

}

<i>theorem(1)</i>	
If $x \neq y$ then $x' \neq y'$.	
<i>proof(1)</i>	
Assume that $x \neq y$ and $x' = y'$. Then by axiom 4, $x = y$.	

$\sqrt{2}$ is irrational

The PRS constructed from the example proof, “ $\sqrt{2}$ is irrational.”, is shown in Figure 2.

4.2 Accessibility in PRSs

Unlike for DRSs, accessibility for PRSs is not just a relation between PRSs, but also a relation between PRS conditions, and between PRSs and PRS conditions. As in the case of DRSs, accessibility can be defined via a subordination relation:

Let Θ_1 and Θ_2 be PRSs or PRS conditions.³ Θ_1 directly subordinates Θ_2 if

- Θ_1 is a PRS, and Θ_2 is the first condition of Θ_1 , or
- Θ_1 and Θ_2 are PRS conditions of a PRS B, and Θ_1 appears before Θ_2 in the list of conditions of B, or
- Θ_1 is a PRS condition of the form $\neg\Theta_2$, or

³ Θ_1 and Θ_2 should be understood to be occurrences of PRSs or PRS conditions. For example, if a PRS has as its condition list $\langle c_1, c_2, c_1 \rangle$, then c_2 subordinates the second occurrence of c_1 , but not its first occurrence. So strictly speaking it is necessary to speak about occurrences (“this occurrence of c_2 subordinates the second occurrence of c_1 ”).

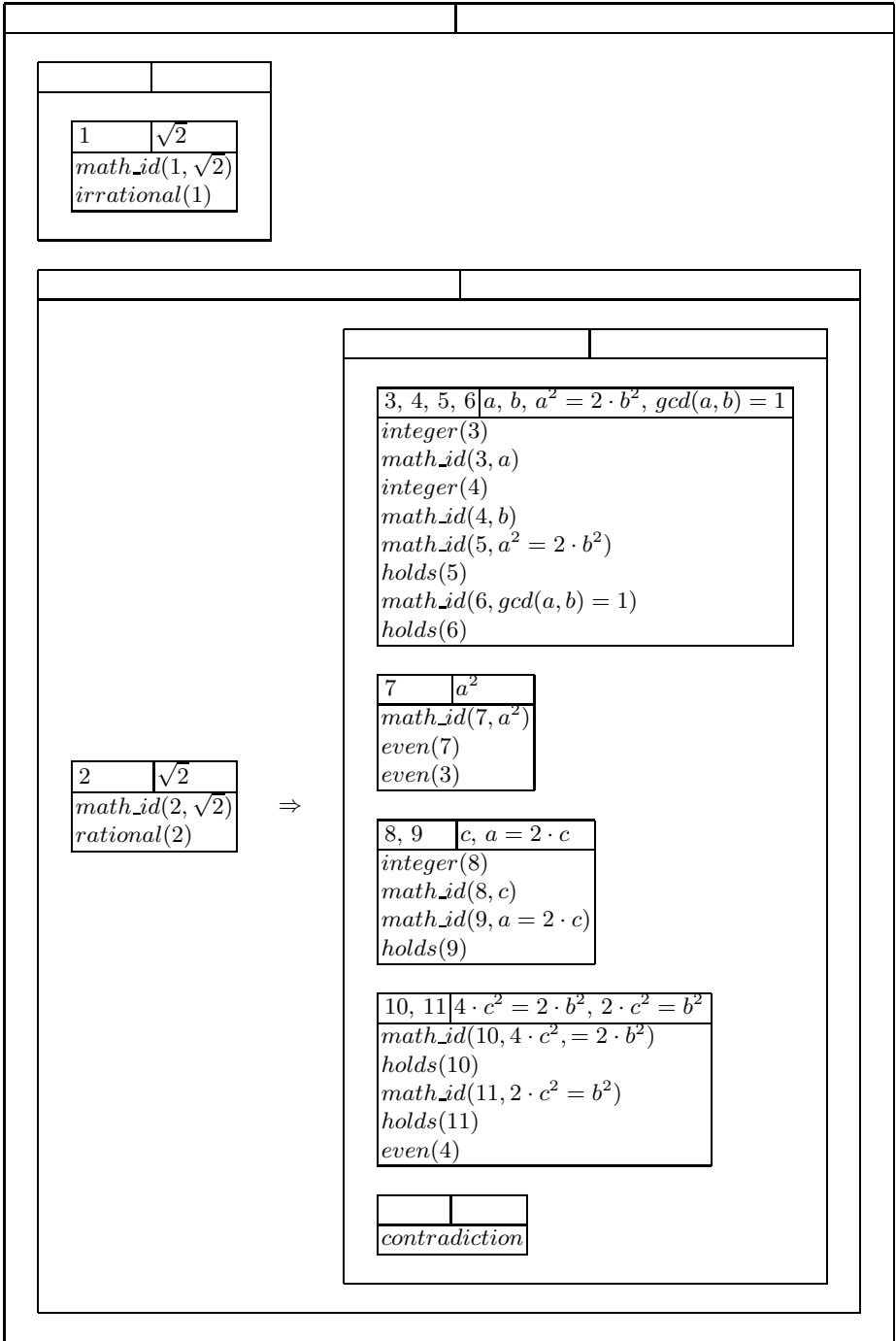


Fig. 2. The PRS corresponding to proof of “ $\sqrt{2}$ is irrational.”, depicted without PRS identifiers and textual references.

- Θ_1 is a PRS condition of the form $\Theta_2 \Rightarrow B$, $\Theta_2 \Leftrightarrow B$, $\Theta_2 \Leftarrow B$, $\Theta_2 \vee B$, $B \vee \Theta_2$, $>< (\Theta_2, B_1, \dots, B_n)$, $<> (\Theta_2, B_1, \dots, B_n)$, $\Theta_2 := B$ or $f : \Theta_2 \rightarrow B$, or
- $\Theta_1 \Rightarrow \Theta_2$, $\Theta_1 \Leftrightarrow \Theta_2$, $\Theta_1 \Leftarrow \Theta_2$, $\Theta_1 := \Theta_2$ or $f : \Theta_1 \rightarrow \Theta_2$ is a PRS condition, or

The relation “ Θ_1 is accessible from Θ_2 ” is the transitive closure of the relation “ Θ_1 subordinates Θ_2 ”.

A discourse referent d or a mathematical referent m is said to be accessible from Θ , if there is a PRS B that is accessible from Θ and that contains d or m in its list of discourse or mathematical referents.

In the PRS of the $\sqrt{2}$ example (Figure 2), we can derive from the above definitions that the discourse referent 2 is accessible from all the PRSs and PRS conditions inside the PRS that is to the right of the “ \Rightarrow ”. The discourse referent 7 and the mathematical referent a^2 are only accessible in the PRSs and conditions that are graphically below the line where they are introduced. On the other hand the discourse referent 1 is only accessible in the PRS in which it gets introduced, as this PRS does not subordinate any other PRSs.

4.3 Motivations for the Accessibility Constraints

For those complex PRS conditions that are analogous to standard DRS conditions (i.e. $\neg A$, $A \Rightarrow B$ and $A \vee B$), the accessibility constraints are equivalent to the standard accessibility constraints in DRT: In all three cases the discourse referents introduced in A and B are not accessible for the later discourse; in $A \Rightarrow B$ the discourse referents from A are accessible in B , but not so for $A \vee B$.

The fact that discourse referents introduced in one of the argument PRSs of a complex PRS condition should not be accessible after that condition naturally extends to those complex conditions that are specific to PRSs ($A \Leftrightarrow B$, $A \Leftarrow B$, $>< (A_1, \dots, A_n)$, $<> (A_1, \dots, A_n)$, $A := B$, $f : A \rightarrow B$).

For example, after the definition “Define m to be square iff there is an integer n such that $m = n^2$.”, neither n nor m should be accessible. The corresponding PRS condition has the form $A := B$ and can be seen in Figure 3. A introduces a discourse referent for m which is also used in B (since m is used to the right of the “iff”). This example also shows that the discourse referents introduced by A should still be accessible in B , as they are according to our above definition of accessibility.

1	m		2, 3	$n, m = n^2$
$math_id(1, m)$:=	$math_id(2, n)$	
$square(1)$			$math_id(3, m = n^2)$	
			$holds(3)$	

Fig. 3. The PRS condition corresponding to the sentence “Define m to be square iff there is an integer n such that $m = n^2$ ”

Similarly, there are natural examples that show that also in conditions of the form $A \Leftrightarrow B$, $A \Leftarrow B$ and $f : A \rightarrow B$ the discourse referents of A should be accessible in B :

- “A natural number n is even if and only if $n + 1$ is odd.”
- “A natural number is even if it is not odd.”⁴
- “For x a positive real, define \sqrt{x} to be the unique positive real y such that $y^2 = x$.”

The condition $>< (A_1, \dots, A_n)$ expresses an exclusive disjunction, so it is natural that accessibility is blocked between the disjuncts just as in the case of the inclusive disjunction condition $A \vee B$. The similar condition $<> (A_1, \dots, A_n)$ expresses the weaker statement that at most one of the “disjuncts” must hold, but it linguistically behaves very similar to the statement that precisely one of them holds, so that the same accessibility constraints apply.

4.4 PRS Semantics

Similarly to the case of the semantics of first order formulae, defining the semantics of PRSs means fixing the criteria under which a PRS is satisfied in a certain *structure*. A *structure* is a set together with relations and functions defined on the elements of the set.

The difference to the semantics of first order logic is that PRS semantics is defined dynamically⁵: Each PRS condition changes the *context* for the subsequent PRS conditions. A *context* is defined to be a triple consisting of a structure and two kinds of variable assignments⁶, one for discourse referents and one for mathematical referents.

- Any definition condition extends the structure in this context triple, so that it contains the newly defined symbol.
- Any PRS condition that is a simple PRS extends the two assignments in the triple, so that they are also defined on the discourse referents and mathematical referents introduced in that PRS.
- Any other kind of PRS conditions doesn’t actually change the context, but must satisfy the context in which it is interpreted.

For a PRS to be satisfied in a certain structure \mathfrak{A} , it must be possible to sequentially modify the context $[\mathfrak{A}, \emptyset, \emptyset]$ by each condition in the PRS.

⁴ Concerning this “reversed conditional” as well as the biconditional above, we should mention that we do not take these examples to be linguistically analogous to donkey sentences. Indeed, these cases should probably be seen as cases of generic interpretation of “a”, whereas the “a” in the antecedent of donkey sentences is normally not considered to be a case of generic interpretation.

⁵ This dynamic definition of PRS semantics is partly based on the dynamic definition of DRS semantics found in [2].

⁶ A *variable assignment* is a function that assigns elements of the domain of the structure to variables, in our case to discourse referents or mathematical referents.

The details of the PRS semantics can be found in [6]. Note that textual referents are not included in the PRS semantics. They are used in the Logic module of the Naproche system (See 5.3).

4.5 Translating PRSs into First-Order Logic

PRSs, like DRSs, have a canonical translation into first-order logic. This translation is performed in a way similar to the DRS to first-order translation described in the book by BLACKBURN and BOS [2]. For example, in both DRS and PRS translations to first-order logic, the introduction of discourse referents is translated into first-order logic using existential quantifiers, unless the discourse referents are introduced in a DRS/PRS A of a condition of the form $A \Rightarrow B$, in which case a universal quantifier is used in the translation. There are two main differences between the PRS to first order logic translation and DRS to first-order logic translations that need discussion:

- *math_id* conditions are not translated into sub-formulae of the first-order translation. A *math_id* condition, which binds a discourse referent n to a term t , triggers a substitution of n , by t , in the translation of subsequent conditions. A *math_id* condition, which binds a discourse referent n to a formula φ , causes a subsequent *holds*(n) condition to be translated to φ .
- Definition conditions are also not translated into sub-formulae of the translation. Instead, a condition of the form $A := B$ triggers a substitution of the relation symbol it defines by the first-order translation of B in subsequent conditions.

Cramer [6] presents a rigorous definition of this translation as well as a proof that this translation is sound, i.e. that a PRS is satisfied in a structure if and only if its translation into first order logic is satisfied in that structure.

5 Formal Mathematics in Naproche CNL

Currently our primary goal is to use the Naproche CNL in the field of *formal mathematics*.

5.1 Formal Mathematics

Mathematicians usually publish *informal proofs*, i.e. proofs for which there are no formal criteria of correctness. Informal proofs contrast with *formal proofs*, in which only syntactical rules can be applied and hence every logical step is explicit.

Formal mathematics aims at carrying out all of mathematics in a completely formal way, i.e. formalise all mathematical statements in strictly formal languages and carry out all proofs in formal proof calculi.

However, formal proofs tend to be lengthy and difficult to read. This problem can be limited by the use of formal mathematics system: These are computer systems which facilitate the formalisation of mathematical proofs. In prominent

examples of formal mathematics systems like *Mizar* [13] and *Coq* [3], this is achieved by allowing the user to use a more readable formal language and to leave out some of the simpler steps of a derivation, which can be filled in by a computer. However, even proofs written in these systems are still not very readable for an average mathematician, mainly because the syntax is more similar to a programming language than to SFML and because they contain much information that human readers consider redundant.

One possible application of the Naproche CNL is to use it in an even more natural system for formal mathematics. As a prototypical version of such a system, we have developed the *Naproche system*.

5.2 The Naproche System

The Naproche system can parse texts written in the Naproche CNL, build the appropriate PRSs, and check them for correctness using automated theorem provers. Apart from several small examples in group theory and set theory, we reformulated the first chapter of EDMUND LANDAU's *Grundlagen der Analysis* [12] in the Naproche CNL, and automatically checked it for correctness using the Naproche system.

The Naproche system consists of three main modules: The User Interface, the Linguistics module and the Logic module. Figure 4 shows an overview of the architecture of the Naproche system.

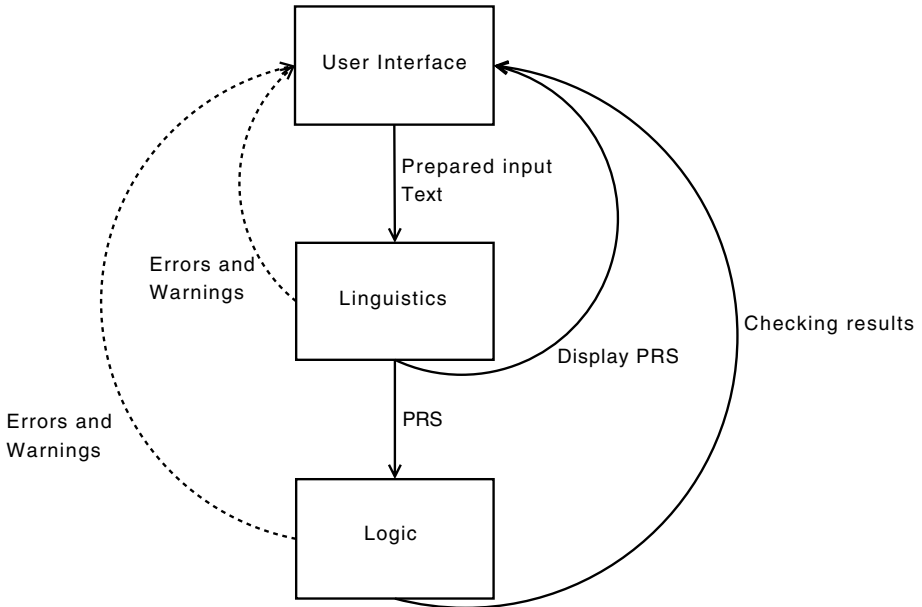


Fig. 4. The architecture of the Naproche system

The User Interface is the standard communication gateway between the user and the Naproche system. The input text is entered using the User Interface, and the other modules report back to the User Interface.

The user enters the input text, written in the Naproche CNL, on the interface, which transforms the input into a Prolog list and hands it over to the Linguistics module. Currently, our User Interface is web-based and can be found on the Naproche website, *www.naproche.net*. We are also working on a plug-in for the WYSIWYG editor TeXmacs [15].

The Linguistics module creates a PRS from the CNL text. The text is parsed using a Prolog-DCG parser that builds up the PRS during the parsing process. The created PRS can either be given to the Logic module, or reported back to the User Interface. With the help of automated theorem provers, the Logic module checks PRSs for logical correctness. The result of the check is sent to the User Interface.

5.3 Checking PRSs

The checking algorithms keeps a list of first order formulae it considers to be true, called the *list of premises*, which gets continuously updated during the checking process.

To check a PRS, the algorithms considers the conditions of the PRS. The conditions are checked sequentially and each condition is checked under the currently active premises. According to the kind of condition, the Naproche system creates obligations which have to be discharged by an automated theorem prover⁷. For example, a *holds* condition is accepted under the premises θ if the automated theorem prover can derive the formula, which is bound to the discourse referent of the *holds* conditions, from θ .

The more premises are available, the harder it gets for the automated theorem prover to discharge an obligation. Therefore we implemented a premises selection algorithm that tries to select only the relevant premises for an obligations. This is where the textual referents of a PRS are used. If a PRS contains a textual referent, the premises corresponding to that referent are taken to be relevant for the corresponding obligation.

If all obligations of a condition can be discharged, then the condition is accepted. After a conditions is accepted, the active premises get updated, and the next condition gets checked. A PRS is accepted by Naproche if all its conditions are accepted consecutively.

If the PRS created from a Naproche CNL text is accepted by our checking algorithm, one can conclude that the text is correct (of course under the assumption that the automatic theorem prover only creates correct proofs).

5.4 Speed and Scalability

One important issue for every algorithm is speed and scalability.

⁷ We access automated theorem provers, currently Otter and E Prover, through GEOFF SUTCLIFF's SystemOnTPTP [14].

Single sentences are parsed in milliseconds, a text of 30 sentences takes about one second. Our biggest example, the Landau text, consists of 326 sentences. The corresponding PRS is created in 11 seconds.

The obligation creation algorithm needs less than a second for small texts (less than 30 sentences), and 4 seconds for the Landau PRS.

Discharging an obligation heavily depends on the automated theorem prover used, the available premises and the actual query. Using the E Prover, version 1.0, all obligations created from the Landau text can be discharged in 80 seconds.

All tests were carried out on a 2 GHz Intel Pentium M with 2 GB Ram.

6 Conclusion

We discussed the particular properties of the Semi-Formal Language of Mathematics and presented the Naproche CNL as a controlled, and thus automatically tractable, subset of SFLM. We defined an extension of Discourse Representation Structures, called Proof Representation Structures, which is adapted so as to represent the relevant content of texts written in the Naproche CNL. Some theoretical properties of Proof Representation Structures were discussed in this paper, but a more exhaustive treatment can be found in the works we referred to.

Since we currently focus on applications in the field of formal mathematics, our implementation, the Naproche system, is geared to the needs of this application. The system translates Naproche CNL texts into Proof Representation Structures, which are then checked for logical correctness; if they are correct, this amounts to the original text being correct.

We believe that the naturalness of the Naproche CNL could make formal mathematics more feasible to the average mathematician.

7 Related and Future Work

CLAUS ZINN developed a system for processing SFLM texts, which he calls *informal mathematical discourse* [17]. He used an extended version of DRs, which he also called Proof Representation Structures, to represent the meaning of a text. The PRS definitions of Naproche and ZINN are different, however.

We are collaborating with the VeriMathDoc project [16], which is developing a mathematical assistant system that naturally supports mathematicians in the development, authoring and publication of mathematical work. We are working towards common formats for exchanging mathematical proofs, e.g. for corpora of mathematical texts which help to assess to what extent Naproche can and should be extended to optimally suit writers-readers of SFML and the purpose of automated verification.

We shall extend the Naproche controlled natural language to include a richer mathematical formula language and more argumentative mathematical phrases and constructs. Once the Naproche CNL is sufficiently extensive and powerful, we shall reformulate and check proofs from various areas of mathematics in a way “understandable” to men and machines.

References

1. Asher, N.: Reference to Abstract Objects in Discourse (1993)
2. Blackburn, P., Bos, J.: Working with Discourse Representation Theory: An Advanced Course in Computational Linguistics (2003)
3. Coq Development Team: The Coq Proof Assistant Reference Manual: Version v8.1 (July 2007), <http://coq.inria.fr>
4. Carl, M., Cramer, M., Kühlwein, D.: Landau in Naproche, ch. 1, <http://www.naproche.net/downloads/2009/landauChapter1.pdf>
5. Cramer, M.: The Controlled Natural Language of Naproche in a nutshell, <http://www.naproche.net/wiki/doku.php?id=dokumentation:language>
6. Cramer, M.: Mathematisch-logische Aspekte von Beweisrepräsentationsstrukturen, Master's thesis, University of Bonn (2008), <http://naproche.net/downloads.shtml>
7. Fuchs, N.E., Höfler, S., Kaljurand, K., Rinaldi, F., Schneider, G.: Attempto Controlled English: A Knowledge Representation Language Readable by Humans and Machines
8. Hardy, G.H., Wright, E.M.: An Introduction to the Theory of Numbers, 4th edn. (1960)
9. Kamp, H., Reyle, U.: From Discourse to Logic: Introduction to Model-theoretic Semantics of Natural Language. Kluwer Academic Publisher, Dordrecht (1993)
10. Kolev, N.: Generating Proof Representation Structures for the Project Naproche, Magister thesis, University of Bonn (2008), <http://naproche.net/downloads.shtml>
11. Kühlwein, D.: A calculus for Proof Representation Structures, Diploma thesis, University of Bonn (2008), <http://naproche.net/downloads.shtml>
12. Landau, E.: Grundlagen der Analysis, 3rd edn. (1960)
13. Matuszewski, R., Rudnicki, P.: Mizar: the first 30 years. Mechanized Mathematics and Its Applications 4(2005) (2005)
14. Sutcliffe, G.: System Description: System on TPTP. In: CADE, pp. 406–410 (2000)
15. Texmacs Editor website: <http://www.texmacs.org/>
16. VeriMathDoc website: <http://www.ags.uni-sb.de/~afiedler/verimathdoc/>
17. Zinn, C.: Understanding Informal Mathematical Discourse, PhD thesis at the University of Erlangen (2004), <http://citeseer.ist.psu.edu/233023.html>

On Designing Controlled Natural Languages for Semantic Annotation

Brian Davis¹, Pradeep Dantuluri¹, Laura Dragan¹, Siegfried Handschuh¹,
and Hamish Cunningham²

¹ Digital Enterprise Research Institute, National University of Ireland, Galway
{brian.davis,pradeep.varma,laura.dragan,siegfried.handschuh}@deri.org

² Sheffield NLP Group, University of Sheffield
hamish@dcs.shef.ac.uk

Abstract. Manual semantic annotation is a complex and arduous task both time-consuming and costly often requiring specialist annotators. (Semi)-automatic annotation tools attempt to ease this process by detecting instances of classes within text and relationships between instances, however their usage often requires knowledge of Natural Language Processing(NLP) or formal ontological descriptions. This challenges researchers to develop user-friendly annotation environments within the knowledge acquisition process. Controlled Natural Languages (CNL)s offer an incentive to the novice user to annotate, while simultaneously authoring, his/her respective documents in a user-friendly manner, yet shielding him/her from the underlying complex knowledge representation formalisms. CNLs have already been successfully applied within the context of ontology authoring, yet very little research has focused on CNLs for semantic annotation. We describe the design and implementation of two approaches to user friendly semantic annotation, based on Controlled Language for Information Extraction tools, which permit non-expert users to semi-automatically both author and annotate meeting minutes and status reports using controlled natural language.

1 Introduction

The Semantic Web endeavors to bring structure to the meaningful content of webpages, creating an environment where software agents can roam freely from web resource to web resource readily carrying out sophisticated tasks for users. Consequently the real power of the Semantic Web will be realised upon the creation of many software agents that collect web content from a range of diverse sources, process the information and exchange the results with other agents. In order for the Semantic Web to become a reality, we need, as a *primer inter pares*, semantic data. The process of providing semantic data is very often referred to as semantic annotation, because it frequently involves the embellishment of existing data, i.e. the text, with semantic metadata, which can subsequently describe the

associated text. Hence, semantic annotation is one of the core challenges for building the Semantic Web and by extension the Semantic Desktop.

1.1 Controlled Natural Languages and Semantic Annotation

Manual semantic annotation can be laborious, time-consuming and costly task, often requiring specialist annotators. (Semi)-automatic annotation tools attempt to ease this process by detecting instances of classes within text and relationships between instances. However, their usage often requires specialist knowledge of Natural Language Processing(NLP), Machine Learning or formal ontological descriptions. This challenges researchers to develop user-friendly annotation environments within the knowledge acquisition process. Controlled Natural Languages (CNL)s offer an incentive to the novice user to annotate, while simultaneously authoring, his/her respective documents in a user-friendly manner, yet shielding him/her from the underlying complex knowledge representation formalisms. “Controlled Natural Languages are subsets of natural language whose grammars and dictionaries have been restricted in order to reduce or eliminate both ambiguity and complexity”[1] The use of CNLs for ontology authoring and population is by no means a new concept and it has already evolved into quite an active research area[2]. However very little research has focused on CNLs for semantic annotation. A natural overlap exists between tools used for both ontology creation and semantic annotation, for instance the Controlled Language for Information Extraction(CLIE)technology permits ontology creation and population by mapping both concept definitions and instances of concepts to a ontological representation using a CNL called CLOnE — Controlled Language for Ontology Editing[3]. However, there is a subtle difference between the process of ontology creation and population and that of semantic annotation. We describe semantic annotation as “a process as well as the outcome of the process. Hence it describes i) the process of addition of semantic data or metadata to the content given an agreed ontology and ii) it describes the semantic data or metadata itself as a result of this process”[4]. Of particular importance here is the notion of the addition or association of semantic data or metadata to *content*.

1.2 Latent Annotation

As with any annotation environment, a major drawback is that in order to create metadata about a document, the author must *first* create the content and *second* annotate the content, in an additional a posteriori, annotation step. In the context of our annotator we seek to merge both authoring and annotation steps into one. This process differs from classic *a-posteriori* annotation resulting in a new type of annotation which we call *latent* annotation. Latent comes from the Latin word with identical spelling who’s etymology is derived from the Latin verb *latere* (lie hidden), a nod in respect to *a-posteriori*(later, what comes after)¹.

¹ <http://www.myetymology.com/latin>

1.3 Controlled Language ANnotation: CLANN and Habitability

This paper describes the design and implementation of **two** user friendly approaches to applying CNL to Semantic Annotation, which we call **CLANN** — **Controlled Language ANnotation**, both of which are based on the CLOnE (Controlled Language for Ontology Editing) language [5]. Both approaches permit non-expert users to semi-automatically both author and annotate meeting minutes and status reports using controlled natural language. **CLANN I** is more automatic and aims to sacrifice expressiveness (wrt the controlling the manipulation and creation of metadata) over usability while in contrast **CLANN II** prioritizes expressiveness (as in control over metadata manipulation) over usability. Uncovering the correct balance between expressiveness and usability is related to the *habitability* problem[6]. A Natural Language Interface(NLI) is considered habitable if users can express everything needed to complete a task, using language they would expect the system to understand. A second aspect of the habitability problem, an aspect which is often overlooked within the CNL community itself, is that of Chomsky’s distinction between competence vs performance [7]. Human linguistic competence can be described as a set of strict rules of a language’s grammar(in this case — English grammar) whereas performance consists of the uses we make of competence . In simpler terms, *how information is written using the grammar* is a measure of competence and *what information could be written using the grammar* is a measure of performance. We argue that the design of CNLs (in the context of knowledge creation) is often driven by competence. The second aspect of habitability however states that an NLI should also attempt to account for both.

The remainder of this paper is organized as follows: Section 2 discusses related work. Section 3 discusses our use case, the target domain, the design and engineering of the domain ontologies and the design and implementation of both CLANN annotators and their corresponding CNLs. Finally, Section 4 offers conclusions, as well as ongoing and future work.

2 Related Work

There are a plethora of tools available for manual or (semi-)automatic semantic annotation, which apply either knowledge based approaches using applied NLP or machine learning techniques or a combination of both to the process. To our knowledge, however, very little research exists involving the application of controlled natural languages to semantic annotation.

The concept of CNL was originally proposed in the 1930s by linguists and scholars who sought to establish a minimal variety of English, in order to make it accessible and usable internationally for as many people as possible (especially non-native speakers)[1]. CNLs were later developed specifically for computational treatment and have subsequently evolved into many variations and flavours such as Smart’s Plain English Program (PEP) [8], White’s International Language for Serving and Maintenance (ILSAM) [8] and Simplified English².

² http://www.simplifiedenglishaecma.org/Simplified_English.htm

They have also found favour in large multi-national corporations, usually within the context of machine translation and machine-aided translation of user documentation [1,8]. The application of CNLs for ontology authoring and instance population is an active research area. *Attempto Controlled English*³ (ACE) [9], is a popular CNL for ontology authoring. It is a subset of standard English designed for knowledge representation and technical specifications, and is constrained to be unambiguously machine-readable into DRS — Discourse Representation Structure, a form of first-order logic. (It can also be re-targeted to other formal languages.) [10]. The Attempto Parsing Engine (APE) consists principally of a definite clause grammar, augmented with features and inheritance and written in Prolog [11]. ACE OWL, a sublanguage of ACE, proposes a means of writing formal, simultaneously human- and machine-readable summaries of scientific papers [12,13].

The Rabbit CNL is a another well known CNL[14]. It is similar to CLOnE in its implementation but is much more powerful with respect to ontology authoring capabilities and expressivity. It has also been favorably evaluated by users in the Ordinance Survey domain but is targeted towards ontology authoring and not semantic annotations. Rabbit has been integrated into Semantic Media Wiki⁴, the purpose of which is to create a user friendly collaborative ontology authoring environment using multiple CNLs[15].

Other related work (in that it creates ABox statements) is WYSIWYM (*What you see is what you meant*)[16]. It involves direct knowledge editing with natural language directed feedback. A domain expert can edit a knowledge based reliably by interacting with natural language menu choices and with the subsequently generated feedback, which can then be extended or re-edited using the menu options. However this process differs substantially from semantic annotation. Similar to WYSIWYM is *GINO* (Guided Input Natural Language Ontology Editor) provides a guided, controlled NLI (natural language interface) for domain-independent ontology editing for the Semantic Web. GINO incrementally parses the input not only to warn the user as soon as possible about errors but also to offer the user (through the GUI) suggested completions of words and sentences—similarly to the “code assist” feature of Eclipse⁵ and other development environments. GINO translates the completed sentence into triples (for altering the ontology) or SPARQL⁶ queries and passes them to the Jena Semantic Web framework. Although the guided interface facilitates input, the sentences are quite verbose and do not allow for aggregation. Static grammar rules exist for the controlled language but in addition, dynamic grammar rules are generated from the Ontology itself as an amendment of additional parsing rules to GINO’s grammar in order to guide the user. This permits the system to

³ <http://www.ifi.unizh.ch/attempto/>

⁴ More information about Semantic MediaWiki can be found at http://semantic-mediawiki.org/wiki/Help:Introduction_to_Semantic_MediaWiki as accessed on 21/06/09

⁵ <http://www.eclipse.org/>

⁶ <http://www.w3.org/TR/rdf-sparql-query/>

handle a domain shift, however this is heavily dependent on any linguistic data or RDF label data encoded the ontology [17]. Finally, [18] presents an Ontology based Controlled Natural Language Editor, similar to GINO, which uses a CFG (Context-free grammar) with lexical dependencies — CFG-DL to generate RDF triples. To our knowledge the system ports only to RDF and does not cater for other Ontology languages.

Futhermore there is the application of controlled natural languages to ontology/knowledgebase querying, which represents a different task to that of knowledge creation and editing but is worth mentioning for completeness sake. Most notably *AquaLog*⁷ is an ontology-driven, portable question-answering (QA) system designed to provide a natural language query interface to semantic mark-up stored in a knowledge base. PowerAqua [19] extends AquaLog, allowing for an open domain question-answering for the semantic web. The system dynamically locates and combines information from multiple domains.

Perhaps the most closely related technology to CLANN is Semantic MediaWiki, which has become a popular way of adding semantics to user generated Wiki pages. A traditional wiki creates links between pages without defining the kind of linkage between pages. Semantic MediaWiki allows a user to define the links semantically, thereby adding meaning to links between pages. Each concept or an instance has a page in the Semantic Wiki, and each outgoing link from this page is annotated with well-defined properties as links. However this kind of approach is not suitable to the kind of semantic annotation that we aim for. The Semantic Media Wiki model forces the users to use the wiki pages for content creation and to create a new page for each instance but does not offer a method to annotate arbitrary text documents which are not intended to be used as wiki pages. Moreover, the relational metadata represented in a Semantic Media Wiki always has the corresponding page as its subject, thereby restricting the creation and description of other relevant entities. Other flavours of Semantic Wikis include IkeWiki⁸ and KiWi.⁹

3 CLANN: Design and Implementation

In this section, we describe our use case, deployment scenarios, the target domain ontology as well as the overall architecture of both CLANN annotators. Each annotator is realized as a GATE pipeline¹⁰ [20]. Finally, we discuss briefly the design and grammars of both types of CLANN CNLs and provide examples.

3.1 A Use Case for Controlled Natural Language for Semantic Annotation

The reader should note that CNLs cannot offer a panacea for manual semantic annotation as a whole since it is unrealistic to expect users to annotate every

⁷ <http://kmi.open.ac.uk/technologies/aqualog/>

⁸ <http://ikewiki.salzburgresearch.at/>

⁹ <http://www.kiwi-project.eu/>

¹⁰ General Architecture for Text Engineering, See <http://gate.ac.uk/>

textual resource using CNL, however there are certain use-cases where CNLs can offer an attractive alternative as a means for manual semantic annotation, particularly in contexts, where controlled vocabulary or terminology is implicit such as health care patient records, business vocabulary and reporting. Our domain use case focuses on project administration tasks such as taking minutes during a project team meeting and writing weekly status reports. Very often such note taking tasks can be repetitive and boring. In our scenario the user is a member of a research group which in turn is part of an integrated EU research project. We choose this domain because: (1) we observed the size of meeting minutes and status reports was quite limited and in addition sentences within such artifacts tended to be short, repetitious and more importantly tended to follow a subject, predicate object pattern, making them good candidates from controlled language information extraction, and (2) we were able to construct our own small corpus of real-world meeting minutes and status reports generated over the three year period from the Nepomuk¹¹ project.

Based on pre-defined templates, the user *simultaneously authors and annotates* his/her meeting minutes or status reports in CNL, using a semantic note taking tool — SemNotes¹², which is an application available for Nepomuk-KDE¹³ — the KDE instance of the Social Semantic Desktop. The metadata is available for immediate use after creation for querying and aggregation. The scenario is not limited to the KDE Desktop or the semantic desktop. Other scenarios involve using Nepomuk Lite — a lightweight version of the platform independent OSGI based Nepomuk Implementation, using a Google Web Toolkit¹⁴ based interface. For the purposes of this paper, we will focus on the Nepomuk-KDE/SemNotes scenario.

3.1.1 Nepomuk-KDE

Nepomuk-KDE is the KDE instance of the EU funded Integrated Project Nepomuk¹⁵, which aims to provide a full implementation of the standards and APIs defined in Nepomuk on the KDE Desktop. The (Social) Semantic Desktop envisioned a new type of collaboration infrastructure intersecting with research across the Semantic Web, Peer-to-Peer (P2P) Networks, and Online Social Networking, culminating in a “novel collaborative environment, enabling the creation, sharing and deployment of data and metadata”[21]. The Nepomuk project provided a specification for a social semantic desktop framework. Nepomuk has continued beyond the lifespan of the project in the form of the OSCAF¹⁶ foundation which aims to foster interoperability between different implementations and publish standards within the community established around the original Nepomuk project.

¹¹ <http://www.semanticdesktop.org>

¹² <http://smile.deri.ie/projects/semm>

¹³ <http://nepomuk.kde.org/>

¹⁴ <http://code.google.com/webtoolkit/>

¹⁵ [http://nepomuk.semanticdesktop.org\(08/01/2010\)](http://nepomuk.semanticdesktop.org(08/01/2010))

¹⁶ <http://www.oscaf.org>

3.1.2 SemNotes

SemNotes is a desktop-based tool for note-taking. It is implemented for the K Desktop Environment ¹⁷ for Linux and uses the Nepomuk-KDE implementation of the Semantic Desktop. The architecture is plugin-based and extensible, so new plugins can be easily added to the application. SemNotes is a semantic application because it uses the ontologies developed in the Nepomuk project ¹⁸ to define its data structures ¹⁹. The vast majority of notes are short and contain only text, hence it was decided that not only the metadata about the notes should be stored in the RDF repository, but also the notes themselves. For storage, SemNotes uses the RDF repository provided by Nepomuk. An important feature of the application is the linking of resources mentioned in text, to the notes. As an application for the Semantic Desktop, SemNotes has direct access to all the resources (like people, places, projects, etc.) available in the local RDF repository. This allows us to query the repository while the user is typing and identify references in text. The data that can be linked to the notes depends greatly on the data available on the desktop as well as on the ontologies that are loaded in the system. In addition, newly discovered relations are stored in the central repository and thus can be accessed by all semantically-aware applications, not just SemNotes. The user benefits from improved search and filtering of notes, easy contextual browsing, as well as better interlinking of data and discovery of new information from that available on his/her computer.

The SemNotes plugins currently available include: a tag cloud, a timeline and a list of linked resources for visualizing metadata about the notes; export and import to and from files for backup of individual notes as well as a RDF exporter plugin for bulk backup. In addition to these, we provide a plugin that employs keyword extraction ²⁰ plugin that allows the user to tag the note with the extracted keywords — the most relevant being suggested as tags. We intend to wrap CLANN technology as a plugin for SemNotes. An overview of SemNotes can be seen below (See Figure 1).

3.2 CLANN I and II Overview

As mentioned earlier, this work focuses on two approaches to Controlled Languages ANnotation — CLANN. Both types of CLANN annotators are implemented in GATE and build on the existing advantages of the CLOnE[5] software and its input controlled language. They share the common features below:

1. Both annotators require only one interpreter or runtime environment — the Java 1.5 JRE.
2. As far as possible, CLANN I and CLANN II are grammatically lax; in particular it does not matter whether the input is singular or plural (or even in grammatical agreement).

¹⁷ <http://kde.org/> (27/04/2009)

¹⁸ <http://nepomuk.semanticdesktop.org/ontologies/> (18/03/2009)

¹⁹ The notes are instances of

<http://www.semanticdesktop.org/ontologies/2007/11/01/pimo#Note>

²⁰ <http://smile.deri.ie/projects/keyphrase-extraction> (23/03/2009)

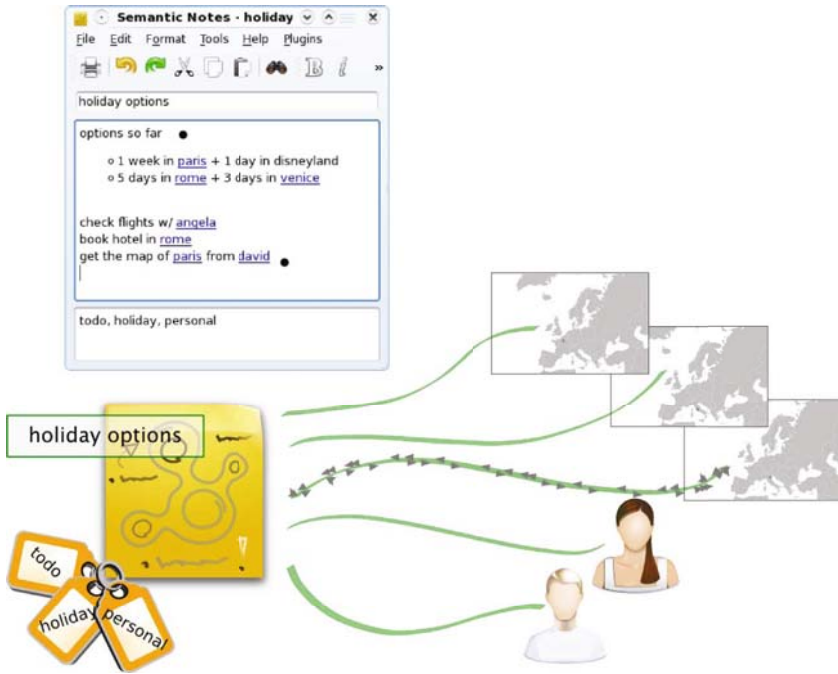


Fig. 1. Overview of SemnNotes

3. Both types of CLANN are compact; the user can create any number of instance properties or instances in one sentence.
4. Both types of CLANN are relatively easy to learn by following examples and a small style guide, without having to study elaborate expressions of formal syntax.
5. Both CLANN annotators are a form of latent semantic annotation — simultaneous authoring and annotation.
6. Both annotators share a common template for meeting minutes.
7. Finally both CLANN annotators share a common GATE Ontology API based on OWLIM²¹. The GATE ontology API can import and export to the following ontology file formats: RDF/XML, N3, NTriples and Turtle.

In our scenario both CLANN are anchored to existing semi-structured data such as a **AgendaTitle**, **Scribe** or **ActionItem** based on predefined meeting minutes or status report templates(see meeting minutes template described below). The

²¹ “OWL in Memory — OWLIM is a high-performance semantic repository developed in Java by Ontotext. The semantics, supported by OWLIM, can be configured through rule-set definition and selection. The most expressive pre-defined rule-set combines unconstrained RDFS with most of OWL Lite”.

See <http://www.ontotext.com/owlim/index.html>(08/01/2010)

templates were constructed based on linguistic introspection and corpus analysis, where conducted over the same collection of status reports and meeting minutes derived from the Nepomuk project.

Template for Type I and Type II annotators

```
Meeting Date:<date>
Project Name:<project name>
Attendees:<attendee1>,<attendee2>
Chair:<chair>
Scribe:<scribe>
```

```
Agenda Items:
Agenda Title:<title>
Comment:<comment>.+
```

```
RoundTable:
Comment:<comment>.+
```

3.2.1 CLANN I: Design and Implementation

The annotator architecture contains a standard GATE pipeline(see Figure 2) which includes the following language processing resources: The GATE English tokenizer, the Hepple POS tagger, a morphological analyser, a finite state lookup gazetteer list component for recognizing useful key-phrases, such as structured elements from the templates and reserved CNL phrases. Any sentences for example, preceded by a **Comment**: element are considered candidates for controlled language parsing. Any remaining tokens from the CNL sentence which are not recognized as reserved CNL key-phrases are used as names to generate links to ontological objects. This is followed by a standard Named Entity(NE) transducer in order to recognize useful NEs, a **preprocessing** JAPE²² finite state transducer(FST) for identifying quoted strings, chunking Noun Phrases(NPs) and additional preprocessing. A second gazetteer list lookup is applied to identify trigger phrases associated with NEs which intersect with quoted and unquoted NP annotation spans. Additional feature values are then added to the NP chunks to indicate the appropriate class to link an NP chunk as an instance to. The last FST parses the CNL from the text and generates the metadata. The current tool is bootstrapped via the Nepomuk Core Ontologies²³ and currently the application creates/populates a meeting minutes/status report ontology MEMO²⁴, which references the users Personal Information Model Ontology(PIMO)²⁵, via the GATE Ontology API. Each meeting minute note should follow a pre-defined

²² Java Annotations Pattern Engine.

²³ [http://www.semanticdesktop.org/ontologies/\(08/01/2010\)](http://www.semanticdesktop.org/ontologies/(08/01/2010))

²⁴ <http://ontologies.smile.deri.ie/2009/02/27/docs/>

²⁵ <http://www.semanticdesktop.org/ontologies/2007/11/01/pimo/>

template below. The template is parsed initially to extract the inherent metadata about the meeting.

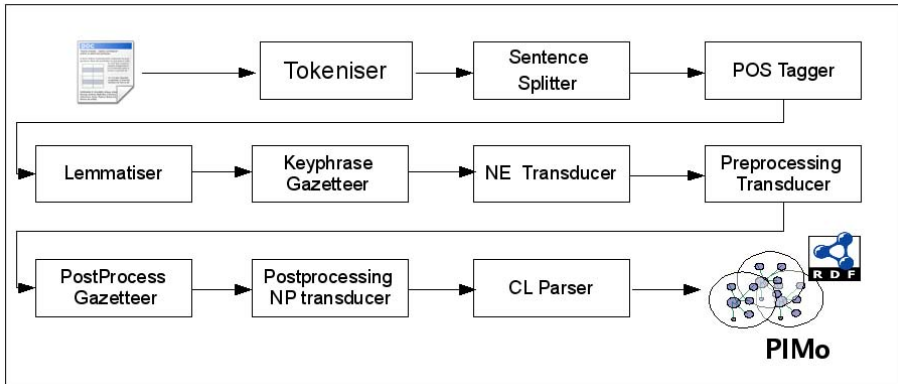


Fig. 2. CLANN I pipeline

Each valid sentence in CLANN I matches exactly one syntactic rule and as mentioned earlier consists of reserved keyphrases (verb phrases, fixed expressions and punctuation marks) as well as *chunks* (which similar to noun phrases are used to name instances). Similar to CLOnE, the language has *quoted chunks*, a series of words which are enclosed in quotes (Eg. “the PhD proposal”). Quoted chunks permit the capture of multi-word expressions as instances. They also permit the use of reserved words that would otherwise be detected by the reserved gazetteer lookup.

An example syntactic rule is contained below:

– $\langle \text{NP} \rangle \langle \text{VP} \rangle (\langle \text{PREP} \rangle ? \langle \text{NP} \rangle) +$

where $\langle \text{NP} \rangle$ corresponds to *chunk* or *Quoted chunk* and $\langle \text{VP} \rangle$ corresponds to reserved verb phrases and paraphrases derived from corpus analysis. Furthermore $\langle \text{PREP} \rangle$ corresponds to any preposition annotated using the POS tagger. Finally $(\langle \text{PREP} \rangle ? \langle \text{NP} \rangle) +$ matches one or more prepositional adjuncts i.e. “for the EU” or “in Work Package 3000”. Hence the above rule would match the following sentences.

Comment: Marco to visit “University of Karlsruhe”.

Comment: Dirk to complete paper by “Sunday 21st June” for “International Semantic Web Conference”.

The above rule extracts the instances as arguments. The reader should note that prior to this stage that standard NE transducer and post-processing NP transducer (see Figure 2) will have collected additional information about each

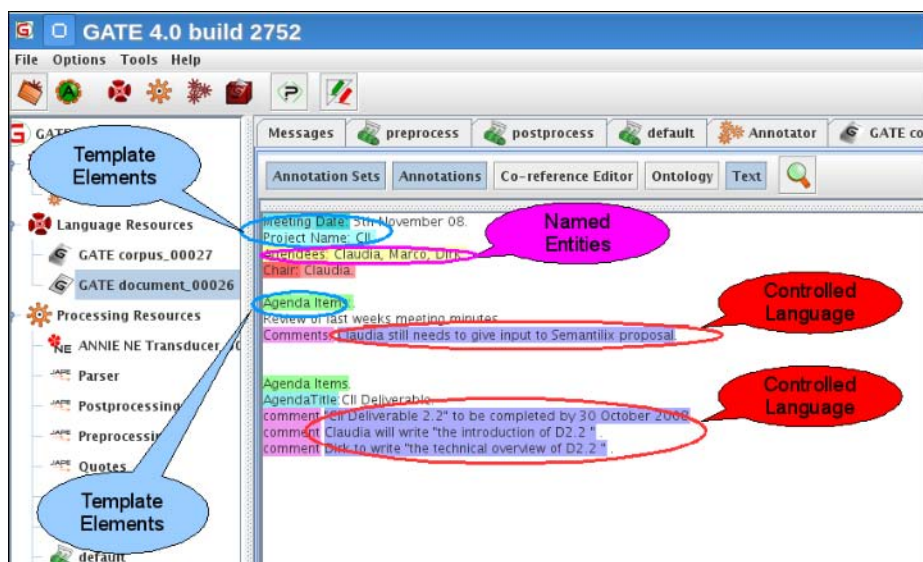


Fig. 3. CLANN I visualised in GATE

chunk. So Marco and Dirk are associated to a Pimo Person, while ‘‘Sunday 21st June’’ will be recognized as a Date. Using similar techniques ‘‘University of Karlsruhe’’ would be recognized as an **Organization** and ‘‘International Semantic Web Conference’’ would be recognized as a **Conference**. The verb phrases to visit and to complete are then used to identify the relevant properties to link the instances recognized.

Other features of CLANN I include nominal co-reference using the Alias: rule, which allows the user to express the same instance in varied forms. It also enables the usage of a *shorthand* by the user when taking minutes.

CLANN I also incorporates elements of language performance in order to make the controlled language more habitable. See Section 1.3 for a discussion on the term habitability. In order to engineer elements of linguistic performance into CLANN I we applied corpus analysis to generate lexicalisations for common properties within the MEMO ontology. These were then added to the the gazetteer list component within the CLANN I pipeline as CNL reserved phrases. We applied mutual information statistics to word frequencies within our corpus to assess the strength of collocational relationships within text. We used the output to generate lists of common trigger phrases which could be aligned to properties within the MEMO ontology. For example:

Comment: Dirk *to complete* paper by ‘‘Sunday 21st June’’ for ‘‘International Semantic Web Conference’’. can be paraphrased as:

Comment: Dirk *to finish up work on* paper by ‘‘Sunday 21st June’’ for ‘‘International Semantic Web Conference’’. or

Comment: Dirk *to wrap up* paper by “Sunday 21st June” for “International Semantic Web Conference”. where **to finish up work on** and **to wrap up** are lexicalisations of the property *toComplete*. On initial inspection, the reader may inclined to view such paraphrases as grammatically incorrect or stylistically inelegant, but recall that CLANN I is language performance driven and seeks to ease the user experience by incorporating such lexicalisations within the controlled language thus making it more habitable. For additional examples of the CLANN I language and grammar, we refer the reader to Table 1.

Table 1. Excerpt of CLANN I grammar with examples

Sentence Pattern	Example	Parsed pattern
<NP><VP><PP>+	Ambrosio to submit "her PhD Proposal" during "the next week".	(Ambrosia <NP>) (to submit <VP>) ((her PhD Proposal <NP><PP>) (during (the next week <NP><PP>)).
	Dirk to work on "the E-Health Proposal" with Ambrosia	(Dirk <NP>) (to work <VP>) (on(the E-Health Proposal<NP><PP>) (with (Ambrosia <NP><PP>)).
ALIAS:<TEXT>;<ALIAS>	Alias:<"CII Deliverable 6.7">;<"D6.7">	Creates “D6.7” as an alias for “CII Deliverable 6.7”.

3.2.2 CLANN II: Design and Implementation

The CLANN II architecture (see Figure 4) is similar in design to CLANN I in that it shares the same language processing resources for tokenisation, sentence splitting, POS tagging and morphological analysis. CLANN II uses an identical template as CLANN I, however the **Comment:** element is non-existent and furthermore sentences themselves are not written in controlled language. In CLANN II the user can write any sentence without restriction under the heading of an **Agenda Item**. What differs in CLANN II is that the user can use snippets of controlled language to associate metadata to a particular piece of text. Snippets of CNL are identified within square brackets using [...]. The CLANN II CNL snippets themselves are similar to CLONE with minor changes. A **preprocessing** finite state transducer (FST) similar to CLANN I is applied to extract values associated with template elements. In addition, text associated to the CNL snippets is also parsed at this stage. The final stage in the pipeline consists of a JAPE transducer which pulls the instances and properties to parse triples, ignoring the unassociated text. CLANN II shares the same GATE API

with respect to ontology manipulation as CLANN I and consults the ontology in similar manner. Similar to CLOnE and CLANN I, the language in CLANN II has *quoted chunks*, a series of words which are enclosed in quotes (“...”). This allows the user to associate metadata to more than one word. Example syntactic rules are shown below:

<TEXT>[IS A CLASSNAME]

where [IS A CLASSNAME] corresponds to a snippet of CNL. Hence:

Dirk[is a Person] to complete paper by “Sunday 21st June”[is a Date] for “International Semantic Web Conference”[is a Conference].
[CHUNK PROPERTY CHUNK]

The above rule allows the user to simply embed a sentence in CNL in order to create relation metadata. This approach also allows users to handle adjuncts with much greater ease. such as associating the Date instance ‘‘Sunday 21st June’’ with *paper*:

[“to complete” same as toComplete]
Dirk[is a Person] to complete paper[is a Document] by “Sunday 21st June”[is a Date] for “International Semantic Web Conference”[is a Conference].
[Dirk “to complete” paper].
[Paper hasEndDate “Sunday 21st June”].

Note, that when creating instances of properties, the controlled language will recognize pre-existing annotations i.e. *paper* and ‘‘Sunday 21st June’’. In order to use a property in the CLANN II CNL, the user must either use the appropriate label for the property on inspection of the ontology (in this case *toComplete* is a part of the ontology) or alternatively they can use the alias preprocessing command to create a more natural substitute for the property.

Another major difference between CLANN I and CLANN II is that the user can also create and manipulate classes, subclasses and class properties. Suppose the user is unsatisfied with the association of *paper* to *Document* and would prefer to associate the text to instance of a non existent class *ConferencePaper*. CLANN II permits the creation of new classes on an ad hoc basis using the following rules:

[<CHUNK> IS A SUBCLASS OF <CLASSNAME>]

resulting in the following:

[“Conference Paper” is a subclass of Document]
Dirk[is a Person] to complete paper[is a “Conference Paper”] by “Sunday 21st June”[is a Date] for “International Semantic Web Conference”[is a Conference].
[Dirk toComplete paper].

We refer the reader to Table 2 for further examples of the CLANN II language.

Table 2. Excerpt of CNL Type II grammar with examples

Sentence Pattern	Example
<TEXT>[IS A <CLASS>].	Dirk[is a Person] Creates an object of the class <i>Person</i> with label <i>Dirk</i> . Note the the label is taken from the document content.
<TEXT>[IS A SUBCLASS OF <CLASS>].	Proposal[is a subclass of Document] or [Proposal is a subtype of Document] Creates a new class with label <i>Proposal</i> as a subclass of the class <i>Document</i> .
<TEXT>[<PROPERTY> <OBJECT>].	Dirk[toComplete "PhD Proposal"] or [Dirk toComplete "PhD Proposal"] Creates a RDF statement which links the instances of <i>Dirk</i> and <i>PhD Proposal</i> with the property <i>toComplete</i> .

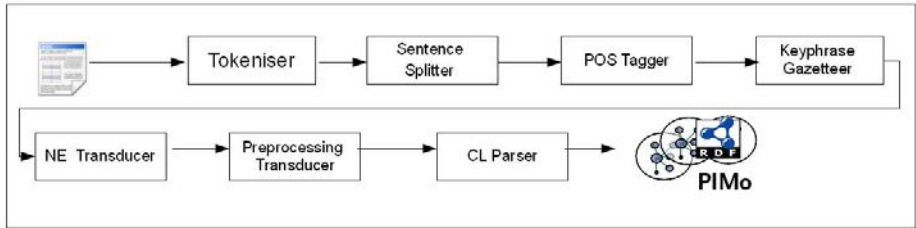


Fig. 4. CLANN II pipeline

3.2.3 CLANN I Vs CLANN II

Both CLANN annotators differ in the following ways:

- The CLANN II language is portable, while CLANN I must be re-targeted for a new domain.
- CLANN I incorporates linguistic performance whereas CLANN II does not.
- CLANN I cannot create TBox statements. This option is available to CLANN II users. This raises interesting research questions: Should a proper annotation tool permit users to alter the ontology at the class level etc? Would annotators then invariably corrupt the ontology?
- There is a subtle difference between annotation in CLANN I and that of typical instance population in other CNLs. This is because *content* is *independent* of the knowledge creation process. In other words, the meeting minute is a human-readable account of a meeting and not merely CNL input for instance creation. In CLANN II however, the CNL acts as the “glue” between free text and the knowledge base. Therefore, CLANN II can be applied to a legacy text which is not the case for CLANN I.

3.3 Domain Ontology

Both CLANN tools are bootstrapped via the Nepomuk Core Ontologies. Our target domain is modeled using a meeting minutes/status report ontology MEMO, which references the users Personal Information Model Ontology(PIMO). The MEMO ontology was initially engineering for purposes of building both CLANN prototypes and was designed as a proof of concept. Since then we have begun to completely redesign and engineer our domain ontology in accordance with proper ontology engineering methodologies, specifically the METHONTOLOGY methodology which outlines a “set of activities that conform the ontology development process” and “a life cycle to build ontologies based on evolving prototypes”[22]. It is essentially a well-structured methodology that the user should apply when building an ontology from scratch. Activities which the ontology engineer should engage in, include: the creation of competency questions as well an ontology requirements specification (ORS) document(See Figure 5). It recommends a series of knowledge acquisition techniques to derive ontological terms including formal text analysis. We applied this using a corpus of status reports and meeting minutes which were collected over the the three year duration. We used Word Smith Tools 5.0²⁶ to conduct corpus analysis in order to extract candidate ontological terms. The new project administration ontology reuses portions of MEMO in line with METHONTOLOGY guidelines and is still work in progress. Figure 5 provides a current overview of the state of the new ontology for meeting minutes and status reports while Figure 6 provides a excerpt of the current iteration of the ORS document.

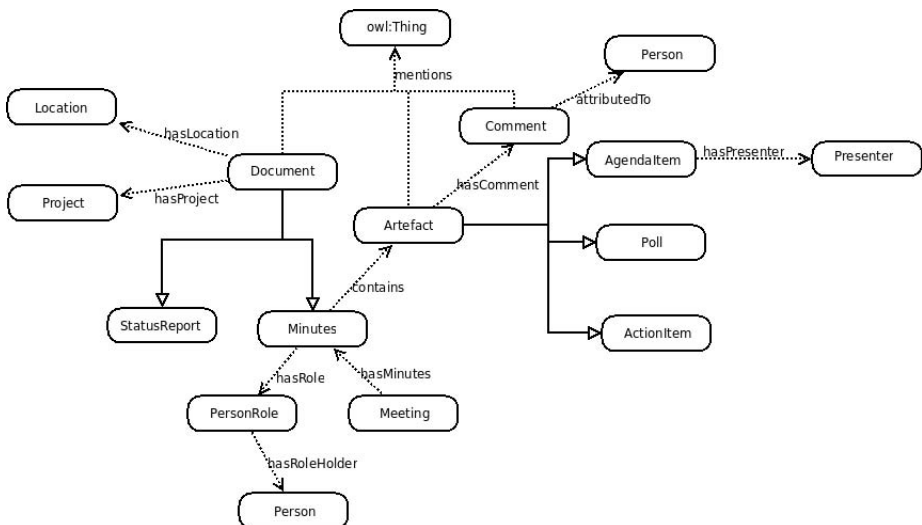


Fig. 5. Current iteration of Domain Ontology

²⁶ <http://www.lexically.net/wordsmith/version5/index.html>

<h2 style="text-align: center;">Project Documents Ontology</h2> <h3 style="text-align: center;">ORS(Ontology Requirements Specification) Document</h3>		
Version: 1.0 Alpha	Scope: <ul style="list-style-type: none"> - meta-data of the meeting itself: location, date, group name, company name, etc - information about various people attending the meeting: Persons, metadata about persons, person rules, etc - information about things presented in the meeting: Agenda's, Round-tables, Discussions, brainstorming, etc - various kinds of discussions: tasks assigned, status reports, announcements, etc - modeling of tasks: start end times, assigned to, task description, sub tasks, task dependency, - modeling of status reports: - modeling the actual content of the meeting: conferences, bugs, documents, deliverables, travel, holidays, projects, hardware, software, etc 	
Ontology Name: Project Document Ontology (PDO) http://smile.deri.ie/ontologies/pdo	Why it Includes -- modelling various project documents like meeting minutes, status reports, final reports, deliverables, etc. The modelling includes <ul style="list-style-type: none"> the general meta data of the documents using Dublin core, specific metadata for specific documents currently only meeting minutes and status reports, support for other project documents may follow later enabling documents to link to the content using annotations (not just tags) 	
Domain: Content and structure of meeting minutes, status reports and other project-specific documents	Why it does not include -- Some of the things that this ontology does not do: <ul style="list-style-type: none"> This ontology does not model the domain of a meeting, instead it focusses on the concepts and relationships of the meeting minutes document. <ul style="list-style-type: none"> Why not model the meeting itself? We don't "need" to model meetings, we deal with the structure of the meeting minutes document The word "Project" in the title specifies a general academic or industrial project, and not the more specific concept of a software project (which is modelled by DOAP) 	
Date: 14th, September, 09.		
Conceptualized by: Pradeep Vamra D., Brian Davis		
Purpose: The ontology will be used for extraction and management of the semi structured knowledge inherent in various formats of project-specific documents like meeting minutes and status reports, and to provide an interesting layer on top of this knowledge for better information retrieval. The information in the documents could be extracted, instantiated against the ontology and stored in an RDF store for further inferencing.		
Level of Formality: Formal - RDFS and OWL - DL		

Fig. 6. Ontology Requirements Specification Document

4 Conclusion and Ongoing Work

Incentive for the user to annotate his/her respective documents plays an important role for the realisation of both the Semantic Web and Social Semantic Desktop. We have described the design and implementation of two approaches to **CLANN** — **Controlled Language for Annotation**, both of which allow non expert users to simultaneously create content within, and add relational metadata to, notes on the Semantic Desktop using CNL. In addition we have described the methodology applied for our ongoing ontology engineering efforts with respect to the domain ontology for meeting minutes and status reports. We intend to finalise the domain ontology and publish in accordance with W3C standards.

As previously mentioned **CLANN I** is more automatic and aims to sacrifice expressivity (wrt the controlling the manipulation and creation of metadata) over usability while in contrast **CLANN II** prioritizes expressivity,(as in control over metadata manipulation), over usability. Uncovering the correct balance between expressivity and usability is related to the *habitability* problem[6]. and NLI is considered habitable if users can express everything needed to complete a task using language they would expect the system to understand. We intend to evaluate the user-friendliness of both CLANN annotators based on the previously successful empirical methods employed in CLOnE [3]. The main research goals of this evaluation will be to(1)the measure and compare the usability of two approaches of CLANN (2)to assess the effects of the respective trade-off between habitability and expressiveness for both CLANN I and CLANN II and(3) to assess whether for certain scenarios controlled languages for semantic annotation can effectively substitute for standard manual semantic annotation tools. Based on the outcome of the evaluation we will hybridize CLANN I and CLANN II into CLANN III for optimal usability and expressiveness (with respect to metadata creation). Finally, we intend to wrap CLANN III as a plugin for SemNotes and complete integration with Nepomuk KDE.

Acknowledgements

The work presented in this paper has been funded in part by Science Foundation Ireland under Grant No. SFI/08/CE/I1380 (Líon-2).

References

1. Schwitter, R.: Controlled natural languages. Technical report, Centre for Language Technology, Macquarie University (June 2007)
2. Smart, P.R.: Controlled natural languages and the semantic web. Technical report, School of Electronics and Computer Science, University of Southampton (2008) (unpublished)
3. Davis, B., Iqbal, A.A., Funk, A., Tablan, V., Bontcheva, K., Cunningham, H., Handschuh, S.: Roundtrip ontology authoring. In: Sheth, A.P., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T.W., Thirunarayan, K. (eds.) ISWC 2008. LNCS, vol. 5318, pp. 50–65. Springer, Heidelberg (2008)

4. Handschuh, S.: Creating Ontology-based Metadata by Annotation for the Semantic Web. PhD thesis (2005)
5. Funk, A., Tablan, V., Bontcheva, K., Cunningham, H., Davis, B., Handschuh, S.: Clone: Controlled language for ontology editing. In: Aberer, K., Choi, K.-S., Noy, N., Allemang, D., Lee, K.-I., Nixon, L.J.B., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudré-Mauroux, P. (eds.) *ASWC 2007 and ISWC 2007*. LNCS, vol. 4825, pp. 142–155. Springer, Heidelberg (2007)
6. Watt, W.C.: Habitability. *American Documentation* 19, 338–351 (1968)
7. Chomsky, N.: *Aspects of the theory of syntax*. MIT Press, Cambridge (1965)
8. Adriaens, G., Schreurs, D.: From COGRAM to ALCOGRAM: Toward a controlled English grammar checker. In: *Conference on Computational Linguistics (COLING 1992)*, Nantes, France, pp. 595–601 (1992)
9. Fuchs, N., Schwitter, R.: Attempto Controlled English (ACE). In: *CLAW 1996: Proceedings of the First International Workshop on Controlled Language Applications*, Leuven, Belgium (1996)
10. Fuchs, N.E., Kaljurand, K., Kuhn, T., Schneider, G., Royer, L., Schröder, M.: Attempto Controlled English and the semantic web. Deliverable I2D7, REWERSE Project (April 2006)
11. Hoefler, S.: The syntax of Attempto Controlled English: An abstract grammar for ACE 4.0. Technical Report ifi-2004.03, Department of Informatics, University of Zurich (2004)
12. Kaljurand, K., Fuchs, N.E.: Bidirectional mapping between OWL DL and Attempto Controlled English. In: *Fourth Workshop on Principles and Practice of Semantic Web Reasoning*, Budva, Montenegro (June 2006)
13. Kuhn, T.: Attempto Controlled English as ontology language. In: Bry, F., Schwerdtel, U. (eds.) *REWERSE Annual Meeting 2006* (March 2006)
14. Dimitrova, V., Denaux, R., Hart, G., Dolbear, C., Holt, I., Cohn, A.: Involving Domain Experts in Authoring OWL Ontologies. In: Sheth, A.P., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T., Thirunarayan, K. (eds.) *ISWC 2008*. LNCS, vol. 5318, pp. 1–16. Springer, Heidelberg (2008)
15. Bao, J., Smart, P., Braines, D., Shadbolt, N.: A controlled natural language interface for semantic media wiki using the rabbit language. In: *Workshop on Controlled Natural Language (CNL 2009)* (March 2009)
16. Power, R., Scott, D., Evans, R.: What you see is what you meant: direct knowledge editings with natural language feedback. In: Prade, H. (ed.) *13th European Conference on Artificial Intelligence (ECAI 1998)*, pp. 677–681. John Wiley and Sons, Chichester (1998)
17. Bernstein, A., Kaufmann, E.: GINO—a guided input natural language ontology editor. In: Cruz, I., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., Aroyo, L.M. (eds.) *ISWC 2006*. LNCS, vol. 4273, pp. 144–157. Springer, Heidelberg (2006)
18. Namgoong, H., Kim, H.: Ontology-based controlled natural language editor using cfg with lexical dependency. In: Aberer, K., Choi, K.-S., Noy, N., Allemang, D., Lee, K.-I., Nixon, L.J.B., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudré-Mauroux, P. (eds.) *ASWC 2007 and ISWC 2007*. LNCS, vol. 4825, pp. 353–366. Springer, Heidelberg (2007)
19. Lopez, V., Motta, E., Uren, V.: Poweraqua: Fishing the semantic web. In: Sure, Y., Domingue, J. (eds.) *ESWC 2006*. LNCS, vol. 4011, pp. 393–410. Springer, Heidelberg (2006)

20. Cunningham, H., Maynard, D., Bontcheva, K., Tablan, V.: GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. In: Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics, ACL 2002 (2002)
21. Decker, S.: The social semantic desktop: Next generation collaboration infrastructure. *Information Services and Use* 26(2) (2006)
22. Fernandez-Lopez, M., Gomez-Perez, A., Juristo, N.: Methontology: from ontological art towards ontological engineering. In: Proceedings of the AAAI 1997 Spring Symposium, Stanford, USA, March 1997, pp. 33–40 (1997)

Development of a Controlled Natural Language Interface for Semantic MediaWiki

Paul R. Smart¹, Jie Bao², Dave Braines³, and Nigel R. Shadbolt¹

¹ School of Electronics and Computer Science, University of Southampton, Southampton,
SO17 1BJ, United Kingdom

{ps02v,nrs}@ecs.soton.ac.uk

² Department of Computer Science, Rensselaer Polytechnic Institute, Troy,
NY 12180, USA

baojie@cs.rpi.edu

³ Emerging Technology Services, IBM United Kingdom Ltd, Hursley Park, Winchester,
Hampshire, SO21 2JN, United Kingdom

dave_braines@uk.ibm.com

Abstract. Semantic wikis support the collaborative creation, editing and utilization of semantically-enriched content, and they may therefore be well-suited to addressing problems associated with the limited availability of high-quality online semantic content. Unfortunately, however, many popular semantic wikis, such as Semantic MediaWiki (SMW), are not sufficiently expressive to support full-scale ontology authoring. Furthermore, the grounding of the Semantic Web in formal logic makes both the comprehension and production of ontological content difficult for many end-users. In order to address these issues, the expressivity of SMW was extended using a combination of semantic templates and a Web Ontology Language (OWL) meta-model. Semantic templates were also used to provide an ontology verbalization capability for SMW using the Rabbit Controlled Natural Language (CNL). The resulting system demonstrates how CNL interfaces can be implemented on top of SMW. The proposed solution introduces no changes to the underlying functionality of the SMW system, and the use of semantic templates as an ontology verbalization solution means that end-users can exploit all the usual features of conventional wiki systems to collaboratively create new CNL verbalization capabilities.

Keywords: Controlled Natural Languages, Semantic Wikis, Rabbit, Semantic Web, Ontology Development, OWL.

1 Introduction

Ever since the early days of the Semantic Web (SW), there have been concerns about the usability of semantic technologies for human end-users [1]. The grounding of the SW in formal logic limits the ability of end-users to participate in the creation, modification and exploitation of semantic content, and unless such usability concerns can be addressed, it is possible that the SW will fail to realize its full potential.

Recently, there have been a number of efforts to improve end-users' ability to create and comprehend SW resources by using natural language (NL) interfaces. As part of their analysis into the kinds of problems encountered by users when working with the Web Ontology Language (OWL), Rector et al [2] identified the need for a "pedantic but explicit paraphrase language" – a language that would support user comprehension by substituting (or at least supplementing) formal logic expressions with NL glosses. Subsequent research has yielded a number of candidate solutions designed to bridge the SW usability gap via NL interfaces. Aside from the development of Manchester OWL Syntax [1], and the attempts of some ontology editing tools to provide NL verbalization solutions [3], members of the Controlled Natural Language (CNL) community have proposed a number of languages to assist users with the creation, modification and exploitation of SW resources (see [4] for a recent review). CNLs are a subset of NL that impose restrictions on both the generation and interpretation of NL expressions [4]. As a subset of NL, CNLs are ideally poised to address the SW usability gap: they capitalize on all the comprehension and productivity benefits of NL, without necessarily undermining the potential for machine-based processing of SW content.

Usability concerns are, however, not the only problem for the SW community when it comes to ensuring the future uptake of SW technologies. Further worries relate to the visibility of practical benefits, the dynamics of conceptual change in specific communities of interest (see [5]), and the mismatch between end-user representational requirements and the nature of available ontological content. Semantic wikis present a potential solution to (at least some of) these problems (see Section 2). They capitalize on the availability of Web 2.0 technologies (which have proved very popular in terms of promoting the large-scale participation of user communities in the generation of online content), and they also avail themselves of opportunities to collaboratively create, edit and exploit semantically-enriched content. Given the ability of wikis to support the collaborative creation of online content, and given the apparent suitability of CNLs as an interface language for the SW, it is possible that a combination of semantic wikis and CNLs could be used to good effect in terms of promoting the greater availability of high-quality online semantic content.

In this paper, we present a semantic wiki system that avails itself of CNL-based ontology verbalization capabilities. The system is called the WikiOnt-CNL system¹, and it is implemented on top of Semantic MediaWiki (SMW) [6], which is one of the most popular and mature semantic wikis currently available. The CNL we focus on, for the purposes of the current paper, is the Rabbit language [7, 8], developed by research staff at the Ordnance Survey² of Great Britain.

The structure of the paper is as follows: Section 2 describes background research and ideas relating to the use of both CNLs and semantic wikis; Section 3 provides an overview of closely related work in this area, specifically the work of Tobias Kuhn to develop a CNL-enabled wiki system using the Attempto Controlled English (ACE) CNL [9, 10]; Section 4 describes the technological infrastructure and capabilities of the WikiOnt-CNL system; Section 5 highlights a number of shortcomings of the WikiOnt-CNL system; Section 6 provides an overview of future work; and, finally,

¹ See http://tw.rpi.edu/proj/cnl/Main_Page

² <http://www.ordnancesurvey.co.uk/oswebsite/>

Section 7 summarizes the key achievements to date and comments on the potential implications of this work for future forms of Web-enabled intelligence.

2 Background: The Semantic Web, Wikis and Controlled Natural Languages

The SW provides a vision of advanced information search, retrieval and processing, made possible by the availability of large bodies of distributed, but heavily inter-linked, data [11]. Although the realization of this vision is now possible, thanks to the efforts of those working in the computer science disciplines, many would argue that we are still waiting for it to become actual. The problem is that the full benefits of the SW are at least partially dependent on the availability of high-quality semantic content, and such content still seems to be in somewhat short supply. In accounting for this dearth of online semantic content, we encounter a number of potential worries and concerns. These include:

- **Poor usability.** SW technologies are often seen as being difficult to learn and use. This imposes a high entrance barrier on those who might otherwise participate in the creation of semantic content.
- **Conceptual mismatches.** There often seems to be a mismatch between the representational requirements of a specific user community and the kind of ontologies that are available for use on the SW. The source of this mismatch may stem from a failure of published ontologies to keep pace with the rate of conceptual change in a specific domain (see [5]), or it may simply stem from the fact that ontology engineers are sometimes far removed from the kind of application domains in which an ontology is ultimately likely to be used.
- **Visibility of practical benefits.** While the practical benefits of the SW are often apparent to those who work in the area of Web technologies, such benefits are often not immediately visible to the wider user community. In particular, it is often difficult for individuals to see the benefits of semantic technologies, either for themselves or for the social collectives in which they live and work.
- **Rewards and incentives.** Finally, there is notion of what we refer to (following psychological research into instrumental and associative conditioning) as the reinforcement schedule associated with the use of semantic technologies. The reinforcement schedule in the case of the SW is, in fact, a complex one; it features elements of both probabilistic and delayed reward and thus might perhaps be best characterized as ‘probabilistic delayed reinforcement’. An alternative (and perhaps more apt) name might be ‘socially-contingent delayed reinforcement’, since the size of the eventual rewards for participating in the SW ultimately depends on the actions of other social agents. Essentially, the idea is that the rewards for semantic technology adoption depend, to a large extent, on what everyone else decides to do; if only a minority of individuals and organizations decide to embrace semantic technologies, then the full benefits of the SW are unlikely to be realized. The reinforcement schedule for adopting semantic technologies is thus somewhat problematic: not only are the rewards temporally displaced, but the actualization of those rewards depends, to a large extent, on the decisions and actions countenanced by other agents.

In thinking about how to approach these issues, it is worth considering that Web users *will* contribute large quantities of online content under at least some circumstances. The first wave of the Web (now commonly referred to as Web 1.0) required users to learn new tools and languages, but many users (even those not well-versed in information technology or computing) still took up the challenge of creating and maintaining online content. With the advent of Web 2.0, the process of adding content became even easier. Web 2.0 websites such as Wikipedia, Flickr, Facebook and Twitter, are all associated with large quantities of user-generated content³, and, what is more, in some cases at least, the user-generated content seems to be of reasonable quality. A recent study of Wikipedia, for example, found that a sample of Wikipedia pages contained content that was approximately equivalent (in terms of its accuracy) to that of corresponding articles in the highly respected Encyclopedia Britannica [12]. What this shows is that some applications can encourage the large-scale participation of users in the creation and editing of online content, and, for the most part, much of this content seems to be of reasonable quality.

One question we can now ask is whether the advent of Web 2.0 software (social software, as it is commonly called) provides us with an effective means of promoting the emergence of the SW (sometimes referred to as Web 3.0). The general idea is that by capitalizing on the availability of popular collaborative content-editing systems (e.g. wikis), we may be able to promote the creation and exploitation of high-quality semantic content (perhaps bypassing or at least attenuating the impact of some of the aforementioned barriers to the adoption of SW technology). The following are some potential benefits of using collaborative content-editing systems, such as wikis:

- **Exploitation of specialist expertise, skills and interests.** Collaborative knowledge editing environments make best use of available end-user abilities. Domain experts can focus on the entry of domain-specific knowledge (often using unconstrained NL), while knowledge engineers can use the expert-contributed knowledge to create more formal knowledge structures.
- **Collective responsibility.** With collaborative editing comes collective responsibility. By enabling all users to contribute information, the boundary between content creator and content consumer is effectively blurred. Users are enabled to become personally involved in the creation and maintenance of information, information that will ultimately be consumed by both themselves and other agents. This radically alters the psychological and sociological significance of online information content.
- **Conceptual change and community consensus.** As has been pointed out by a number of commentators (e.g. [13]), ontologies are not just formal representations of domain knowledge, they are also *community contracts* about the kind of knowledge that is deemed to be important in a domain. This emphasis on community consensus suggests that the current approach to ontology development, one in which a large number of ontologies is developed by a rather select group of ontology engineers, is unlikely to be a viable long-term strategy. Instead, we need to seek a more decentralized and community-based approach to ontology development, one in which knowledge structures emerge (deliberately or otherwise) via the

³ For example, Wikipedia has 3,014,758 English articles as of 27th August 2009. It also has 10,378,900 named user accounts.

collective actions of (perhaps) large numbers of individuals. Another benefit of decentralization is that it enables ontologies to evolve in accord with the conceptual dynamics of a particular user community or epistemic domain.

- **Socially-mediated instant gratification.** Whereas the conventional approach to ontology development encounters problems of socially-contingent delayed reinforcement (see above), the wiki-based approach to creating semantic content introduces the idea of what we might refer to as ‘socially-mediated instant gratification’. This is the idea that the benefits of semantic-enrichment are readily apparent in a semantic wiki environment. We can use semantic annotations to (e.g.) perform sophisticated forms of information retrieval or to automatically assert novel categories of information that are of interest to multiple users (e.g. “all articles about movies directed by Spanish film directors who have directed more than 10 films and who are now deceased”). What is more, these benefits result from the collective actions of large numbers of individuals – in other words, they are socially-mediated. As a result, we encounter a situation in which greater collective effort yields greater individual and collective rewards.

In spite of all these potential benefits, however, the problem concerning the usability of SW technologies remains an important concern. The response of the CNL community to this problem has been to develop a number of CNLs to support ontology development. Thus, we encounter languages like Rabbit [8], Sydney OWL Syntax (SOS) [14] and an OWL-compliant subset of ACE, called ACE-OWL [15]), all of which have been used to develop domain ontologies for the SW. In most cases, the user evaluation studies that have been undertaken with these languages suggests that they support the kind of comprehension and production benefits one would expect to see with NL renditions of formal logic models [8, 16, 17].

All of this leads to a number of specific proposals as to how high-quality semantic content can be made more available on the SW. One proposal concerns the integration of semantic technologies with existing Web 2.0 applications. The aim here is to kick-start the delivery of semantically-enabled (Web 3.0) capabilities by building on the existing platform provided by Web 2.0 technologies⁴. A second proposal concerns the use of CNLs to improve the user experience of creating, consuming, checking and correcting semantic content. Thus, we end up with a seemingly sensible technology development strategy: semantically-enable existing Web 2.0 applications and then extend their user interaction capabilities with one or more CNL interfaces. This is the aim of the research reported here. Our research aims to show how SMW can be adapted to support full scale ontology authoring using an OWL meta-model. It also shows how both ontologies and semantically-annotated wiki content can be serialized to a specific CNL, namely Rabbit, within the context of the SMW environment.

3 Related Work

There have been a number of attempts to develop tools for creating and editing ontologies using CNLs (e.g. Ace View [18] and ROO [19]). In the specific context of

⁴ It should probably be borne in mind, of course, that a similar dependency holds between Web 2.0 and Web 1.0: Web 2.0 capabilities are only possible because they build on Web 1.0 technologies (browsers, protocols, markup languages and all the rest).

applying CNL technology to collaborative Web-based knowledge editing environments (e.g. wikis), the work that is most closely related to the work reported here is that associated with the development of the AceWiki system [9, 10]. There are a number of points of comparison between AceWiki and the current system. The following list provides an overview of some of the key similarities and differences:

- **Wiki system.** AceWiki uses a proprietary wiki system that is built specifically to support content authoring using ACE. WikiOnt-CNL is implemented on top of SMW, which itself is implemented as an extension to the popular MediaWiki engine⁵. WikiOnt-CNL has all the functionality of SMW, and because SMW has an active developer community there are lots of additional extensions that can be used to extend its functionality even further (e.g. semantic map extensions enable users to create, edit and view semantic geospatial data using an inline map view⁶).
- **CNL interface.** AceWiki provides an advanced CNL editor for the ACE language which can be used to create knowledge statements, semantic rules and inline queries. WikiOnt-CNL does not (at present) provide an editing interface for CNLs; its functionality is primarily geared towards the verbalization of existing ontological content using the Rabbit CNL.
- **Semantic expressivity.** AceWiki can support the representation of models whose expressivity is greater than OWL, but it does not (at the present time) support the representation of OWL datatype properties. WikiOnt-CNL provides full support for OWL via the OWL meta-model extensions referred to in Section 4.1. However, WikiOnt-CNL does not provide support for models whose expressivity is greater than OWL (version 1.1)⁷.
- **Rules and reasoning.** AceWiki supports the representation of rules and can perform reasoning on ACE sentences. WikiOnt-CNL can support the representation of certain types of rules (see [20]), but its reasoning capabilities are currently limited.
- **Support for multiple CNLs.** AceWiki, as its name suggests, has been developed specifically to support the collaborative creation of ontologies using the ACE CNL. WikiOnt-CNL currently supports the Rabbit CNL, but its functionality can be extended to accommodate multiple CNLs. Users can create custom CNL verbalization capabilities by (collaboratively) editing the relevant CNL verbalization templates (see Section 4.4).
- **Export capabilities.** AceWiki content can be exported as both ACE texts and OWL ontologies. WikiOnt-CNL content can be exported in any format providing an appropriate export template exists. Users can use one of the existing CNL templates, or they can create their own custom export template.
- **Import capabilities.** AceWiki does not provide an import capability at the present time. WikiOnt-CNL can import OWL ontologies; however, in some cases, the CNL texts generated from an imported ontology are of poor quality. This is

⁵ MediaWiki is the software used by all projects of the Wikimedia Foundation. These projects include Wikipedia, Wiktionary and Wikispecies.

⁶ See http://www.mediawiki.org/wiki/Extension:Semantic_Maps.

⁷ It should also be borne in mind that SMW does not support the full range of simple XML Schema datatypes. A default installation of SMW includes support for string, Boolean, date, text, number, URL, and geographic coordinate datatypes.

because most imported ontologies do not contain the kind of linguistic knowledge that supports the appropriate morphological realization of ontology elements in grammatically-diverse sentential contexts (see Section 5).

Of all these features, the one that is perhaps the most notable is the nature of the editing interface. In the case of AceWiki, a predictive editor interface supports users in the creation and editing of CNL sentences, so it is well poised to capitalize on the putative production benefits of CNLs [16]. Our system, in contrast, does not (at the present time) provide a direct editing interface for CNLs. Rather, it assumes that semantic content will be created by other means, e.g. the addition of typed hyperlinks or the creation of ontologies using a custom form-based editor (see Section 4.3). While the lack of a CNL-based editing interface may seem to be a deficiency of our approach, it is also important to remember that we are attempting to capitalize on the existing mechanisms that SMW provides for the creation of semantic content. Specialist CNL editing interfaces are certainly a desirable addition to the ways in which users can create and edit ontological content; however, we do not want to necessarily limit user interaction to these interfaces. Rather, we want such interfaces to work in concert with existing mechanisms of semantic content creation. AceWiki does not, at the present time, support the kind of source page editing that is common to many wiki systems, and it does not, therefore, allow users to add HTML content to a page and then annotate this content with metadata elements. For us, this is an important point of difference. We see the conventional forms of wiki-based content editing as an essential element of the success of wikis in promoting large-scale collaborative knowledge editing. Firstly, we think that an ability to add unconstrained NL content to a page is an important aspect of the knowledge engineering lifecycle – it enables some contributors (e.g. domain experts) to contribute unstructured information that serves as the source material for more formal knowledge modeling efforts. This approach enables multiple individuals to contribute to the knowledge infrastructure of an application in ways that best suit their idiosyncratic skills, abilities and interests. Secondly, when seen as an attempt to improve the comprehensibility of semantically-enriched resources, it seems important for CNL-enabled semantic wikis to provide a number of ways of visualizing and interacting with semantic content. AceWiki does provide a number of useful capabilities here, including the ability to add comments to specific CNL sentences, an ability to view inferred as well as asserted statements, and an ability to create and execute inline queries. What it does not provide, however, at least at the present time, is an ability to add other types of content that might prove beneficial to a human end-user's understanding of semantic content. By this we mean multimedia content, page history logs, discussion threads, and extended narratives written in plain (unconstrained) NL. SMW, as an extension of the MediaWiki engine, provides all the functionality that is commonly seen in Wikipedia, and it therefore provides access to all of the aforementioned features.

4 The WikiOnt-CNL System

4.1 Overview

In order to combine the benefits of semantic wikis and CNLs, we extended the functionality of SMW to provide full OWL 1.1 authoring and Rabbit CNL verbalization

capabilities. The system comprises four types of components, all of which are built on top of SMW and the MediaWiki engine. The components are:

1. **Semantic Templates:** Semantic templates provide a means by which the text of a wiki page can be programmatically generated by the wiki parser engine. In essence, the wiki parser replaces a template with the text given on the template's source page. The template may be parameterized, such that different kinds of text output are generated by the parser. Furthermore, templates can be replaced with another template so that complex forms of template embedding can occur. In the context of the WikiOnt-CNL system, semantic templates are used to represent ontological content and generate Rabbit CNL sentences.
2. **Semantic Forms:** Semantic forms are an extension to MediaWiki that work in conjunction with semantic templates. Semantic forms enable users to create and edit structured data using conventional web form controls, e.g. text boxes, list boxes, checkboxes and so on. In the context of the WikiOnt-CNL system, semantic forms are used to support the indirect creation and editing of ontology elements (i.e. classes, properties, individuals) via semantic templates (see Section 4.3 for more details).
3. **Special Pages:** Special pages are pages that provide specific services and functionalities to end-users. Within the WikiOnt-CNL system, special pages are used to support ontology import/export capabilities, CNL export capabilities and semantic query capabilities.
4. **Built-In Properties:** Built-in properties are wiki pages that are defined in the "Property:" namespace. They correspond to relationships defined as part of the SMW OWL meta-model (see Section 4.2), and they therefore support the representation of semantic information that surpasses the rather limited semantic expressivity of a default SMW installation.

The key components of the WikiOnt-CNL system are illustrated in Fig 1. The following list summarizes the functionality of the WikiOnt-CNL system with specific reference to these components.

- **WikiOnt-CNL Semantic Templates:** Semantic Templates form the core of the WikiOnt-CNL system. They support the generation of Rabbit CNL Sentences, and they also provide a mechanism to extend the representational expressivity of SMW using the SMW-mOWL meta-model. In accord with these two functions, WikiOnt-CNL Semantic Templates come in two basic flavors: SMW-mOWL Templates and CNL Verbalization Templates. SMW-mOWL Templates encode meta-model information about the ontology element (i.e. OWL Class, OWL Property or OWL Individual) described on a particular Wiki Page (this information is ultimately translated to a set of RDF triples and stored in the SMW database); CNL Verbalization Templates generate Rabbit CNL Sentences by retrieving semantic information from the SMW database and structuring the retrieved information according to the Rabbit syntax specification. Template inclusion and parameterization are handled by the WikiOnt-CNL Semantic Form components.

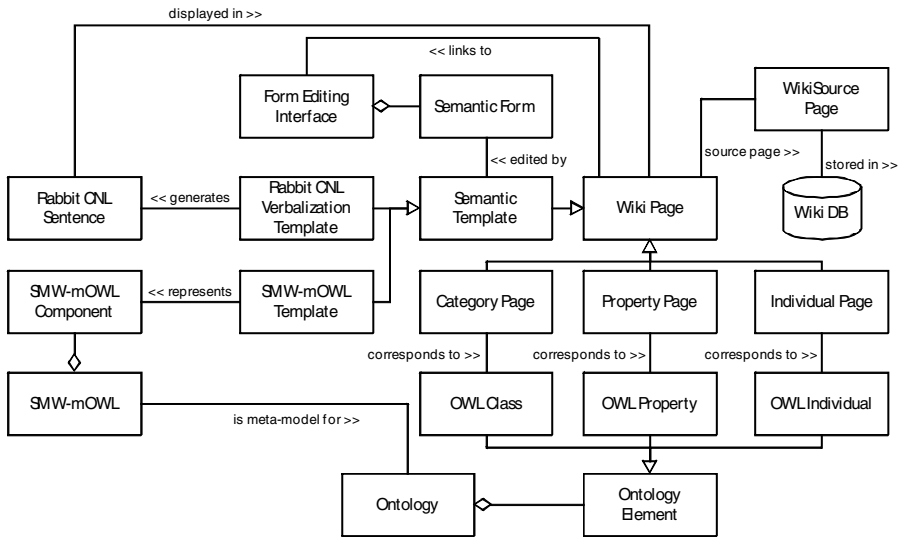


Fig. 1. Main components of the WikiOnt-CNL system. Components not shown in the figure include Special Pages (SPARQL Query, RDF Export, CNL Parser, CNL Export, and Ontology Import pages) and additional interface components (e.g. CNL Editor).

- SMW-mOWL:** SMW-mOWL is a meta-model for OWL ontologies implemented within the WikiOnt-CNL system. It enables OWL 1.1. ontologies to be developed within SMW⁸. SMW-mOWL is implemented within the WikiOnt-CNL system as a collection of built-in properties that encode information about the various relationships and semantic axioms associated with specific ontology elements. For example, the owl:disjointWith axiom is represented in the WikiOnt-CNL system as the “Owl:disjointWith” property, which is declared in the “Property:” namespace. Similarly, a logical conjunction of two classes can be represented using the “Owl:unionOf” property, again declared in the “Property:” namespace. Together, these properties enable the Wiki Pages corresponding to particular Ontology Elements to be annotated in a way that is compatible with existing SMW capabilities. Thus, when asserted on the page “Category:IranianCity”, the semantic annotation “[Owl:disjointWith::Category:AfghanCityl]” asserts that the “IranianCity” class is disjoint with the “AfghanCity” class. When parsed by the Wiki Parser, the annotation will be translated to the triple “<wiki:IranianCity property:disjointWith wiki:AfghanCity>” and stored in the SMW database. In this way, information about the relationships and logical axioms associated with Ontology Elements can be represented within the existing annotation framework of SMW.
- WikiOnt-CNL Semantic Forms:** WikiOnt-CNL Semantic Forms support the creation and editing of Semantic Templates. Because the text in Semantic Templates ultimately gets replaced by wiki-text, Semantic Templates provide a convenient way to simplify the creation and modification of complex page content. Nevertheless, the

⁸ By default, SMW provides only limited support for the representation of OWL constructs.

direct editing of the Semantic Templates included with the WikiOnt-CNL system is too complex for most casual users; it requires users to have a detailed knowledge of the kind of templates that are available and the kind of parameters such templates should be used with. Semantic Forms provide a solution here. They enable users to interact with a form-based interface that hides much of the complexity associated with the entry of structured knowledge content. Within the WikiOnt-CNL system, Semantic Forms are defined within the “Form:” namespace, and they can be invoked from the main interface of a Wiki Page via a hyperlink. Different Semantic Forms are provided for the creation and subsequent editing of classes, properties and individuals.

- **Wiki Pages:** Wiki Pages within the WikiOnt-CNL system correspond to the standard wiki pages that one sees in typical installations of MediaWiki (e.g. Wikipedia) and SMW. Because the WikiOnt-CNL system is built on top of SMW without replacing or eclipsing any of its functionality, a Wiki Page in WikiOnt-CNL can be treated in exactly the same manner as a Wiki Page in SMW. WikiOnt-CNL does not oblige end-users to use either the meta-modeling extensions for ontology authoring; neither does it require end-users to rely on the template-mediated mechanism for CNL verbalization. In fact, because SMW already supports the representation of certain types of semantic information, e.g. taxonomic hierarchies, class membership and relationships between individuals, the Rabbit CNL Verbalization templates can be used independently of the OWL meta-modeling solution described here.
- **WikiOnt-CNL Special Pages:** Special pages within the WikiOnt-CNL system are used to provide functionalities that go beyond those provided by SMW. Five Special Pages are being made available as part of the WikiOnt-CNL system. These include a CNL Parser Page, an OWL Import Page, a CNL Export Page, an RDF/XML Export Page, and a WikiOnt-CNL SPARQL Endpoint. Of these, the OWL Import, RDF/XML Export and SPARQL Endpoint pages provide services that are similar to those already provided by SMW. The reason we have implemented WikiOnt-CNL-specific variants of these services is because the existing services do not take account of the semantic enrichment provided by the SMW-mOWL meta-model. Rather than modify or replace the existing services, we suggest it is more appropriate to provide new services that specifically cater for users interested in exploiting the SMW-mOWL extensions.
- **CNL Editing Interface:** The CNL Editing Interface component constitutes part of our proposed future work on the WikiOnt-CNL system. (see Section 6). The aim, in this case, is to develop an editor that provides end-users with the option of authoring (and editing) ontologies using one or more CNLs.

4.2 OWL Meta-model Extensions to Semantic MediaWiki

As discussed by Kuhn [10], one drawback associated with some semantic wiki systems, concerns their rather limited expressivity. The fact is that many semantic wikis lack the expressivity required for the development and representation of semantically-expressive ontologies. SMW is no exception here. Its modeling capabilities are largely limited to subsumption hierarchies and property specifications. Thus, in order to support the realistic use of SMW as an ontology development environment, we

need to extend its representational capabilities to support the full range of OWL language elements (such extensions are, of course, also important in terms of demonstrating the representational capabilities of OWL-compliant CNLs within SMW).

Within our system, OWL 1.1 authoring capabilities are enabled via the use of semantic templates and an OWL meta-model, called SMW-mOWL (where the ‘m’ stands for meta-model). Individual pages within the wiki correspond to a particular type of ontology element (i.e. class, property or individual), and, within each page, information about the target ontology element is encoded using a collection of meta-model oriented semantic templates. The semantic templates follow the structure of the SMW-mOWL. So, in the case of OWL classes, there are templates to represent both named and anonymous classes, as well as templates to represent the relationship between classes (e.g. `rdfs:subClassOf` and `owl:equivalentClass`). In the case of OWL properties there are two templates: one to represent basic information about a property and one to represent relationships between properties. Finally, in the case of individuals, there are templates to represent information about class membership and the relationships between specific individuals.

Wiki pages corresponding to classes are always created in the “Category:” namespace. This accords with the convention in SMW where “Category:” pages are deemed to represent categories of pages, and the relationship between a category page and its (direct) sub-categories is interpreted as a subsumption relationship. A “Category:” page can include either the “Template:NamedClass” template or the “Template:AnonClass” template (these represent named and anonymous⁹ classes respectively), both of which encode basic information (e.g. text label) about the class. Clearly, a specific class (anonymous or otherwise) needs to be associated with more information than just text labels; therefore a “Category:” page often includes multiple types of SMW-mOWL template (e.g. the “Template:NamedClassRelation” is used to represent `rdfs:subClassOf` relationships).

Properties are created within the “Property:” namespace, again following the convention in SMW. Basic information about a property is encoded using the “Template:Property” template, and relationships between properties (e.g. `rdfs:subPropertyOf`) are encoded using a “Template:PropertyRelation” template.

Instances of classes are represented in pages which are in the main namespace of the wiki (i.e. the page name is not preceded by a colon terminated label, such as “Category:” or “Property:”), or in additional namespaces as specified by the wiki administrator. Relationships between individuals, for example, “Hamid_Karzai isPresidentOf Afghanistan” (where “isPresidentOf” represents an instance of the “Property:IsPresidentOf” property), are always associated with the subject page (i.e. the page corresponding to “Hamid_Karzai” in the case of the current example). Currently, the wiki system only supports individuals that are instantiated from named classes (i.e. “Category:” pages containing the “Template:NamedClass” template). No direct instances of anonymous classes are permitted.

In order to exemplify the use of templates in encoding information about ontology elements, consider the following class expression, as represented in OWL Abstract Syntax:

⁹ An anonymous class is typically an instance of the `owl:Restriction` class. It is automatically assigned a unique number by the WikiOnt-CNL system (e.g. “Category:-1”).

```
Class(AfghanTribalChief partial TribalChief
      restriction(isLeaderOf someValuesFrom(
        Tribe restriction(isLocatedIn Afghanistan))))
```

This class will be represented in the WikiOnt-CNL system using the following set of wiki pages (the names of the pages are given in bold font; the statements between curly braces – `{{...}}` – correspond to specific SMW-mOWL templates).

Category:AfghanTribalChief

```
{{NamedClass |label=Afghan Chief |plural=Afghan Chiefs }}
{{NamedClassRelation |type=subClassOf |class=TribalChief }}
{{SomeValuesFrom |on property=isLeaderOf |on class=-1 }}
```

Category:TribalChief

```
{{NamedClass |label=Tribal Chief |plural=Tribal Chiefs}}
```

Category:Tribe

```
{{NamedClass |label=Tribe |plural=Tribes}}
```

Category:-1 (automatically numbered)

```
{{AnonClass}} {{AnonClassRelation |type=subClassOf
|class=Tribe}} {{HasValue |on property=isLocatedIn
|has value=Afghanistan}}
```

Property:isLeaderOf

```
{{Property |label=is leader of |type=Object}}
```

Property:isLocatedIn

```
{{Property |label=is located in |type=Object}}
```

Afghanistan

```
{{Individual}}
```

The semantic templates associated with each page can be created manually, but we suspect most users will opt to use the form-based interface described in Section 4.3. As with any template built within the MediaWiki environment, the wiki parser replaces the text contained within the curly braces by referencing the syntax of the appropriate template. Thus, if we examine the source of the “Template:SomeValuesFrom” template, we see the following:

```
[[owl:someValuesFrom::Property:{{{on property}}};
Category:{{{on class}}}]]
```

When the “SomeValuesFrom” template on the “Category:AfghanTribalChief” page is parsed by the wiki, it is replaced with the following text:

```
[[owl:someValuesFrom::Property:isLeaderOf;Category:-1| ]]
```

This markup asserts a relationship between the `AfghanTribalChief` class and a text string, corresponding to “Property:isLeaderOf;Category:-1”, via the property “Property:Owl:someValuesFrom” (which is represented as a built-in property of type

‘String’ in the WikiOnt-CNL system). Based on this mechanism of template-mediated text substitution, the logical infrastructure of the ontology is encoded in a form that can be stored in the SMW knowledge base. Importantly, this approach to knowledge representation within SMW does not require any changes to the underlying SMW source code or knowledge base; it simply builds on the existing capabilities of SMW and the MediaWiki engine. Such an approach makes it easy to implement WikiOnt-CNL in existing SMW systems (the editing and CNL verbalization capabilities can be installed simply by creating some special purpose “Property:” pages and copying the semantic templates from the WikiOnt-CNL system to the existing SMW system).

4.3 Ontology Editing Interface

Most users will, we suspect, not relish the prospect of editing the source code of a wiki page containing SMW-mOWL templates. For this reason, we have implemented a number of semantic forms that enable end-users to edit the SMW-mOWL templates using a form-based interface. Semantic forms are an extension to the MediaWiki engine, and they enable developers to support users in entering structured data. The availability of such forms provides us with a means to support the creation and editing of SMW-mOWL templates.

Fig 2. illustrates the editing interface for a specific (named) class (the *AfghanTribalChief* class we encountered earlier). We can see from Fig 2. that the editing interface for classes is composed of multiple sections, each corresponding to a specific SMW-mOWL template. Thus, the ‘Basic Information’ section corresponds to the

Basic Information

Label (Singular):

Afghan Chief

Label (Plural):

Afghan Chiefs

Ontology:

Cultural Ontology

Relation to other classes

AfghanTribalChief is

☐ None

☒ subClassOf

☐ equivalentClass

☐ complementOf

☐ disjointWith

the class

TribalChief

Remove

Add another

The class must have some property values from

Every AfghanTribalChief has some values on the property

IsLeaderOf

to one or more members of the class

-1

Remove

Add another

Fig. 2. The form-based editing interface associated with OWL classes. The specific class being edited in this case is the “Category:AfghanTribalChief” class.

“Template:NamedClass” template and permits the editing of basic information associated with a named class; the ‘Relation to other classes’ section corresponds to the “Template:NamedClassRelation” template and permits the assertion of (e.g.) subclass relationships; and, finally, the ‘The class must have some property values from’ section corresponds to the “Template:SomeValuesFrom” template and enables users to assert information about existential restrictions (in this case, that an “AfghanTribalChief” is the leader of at least one thing that is both a Tribe and is located in Afghanistan). Other sections provide editing interfaces for the other types of SMW-mOWL template.

4.4 Rabbit CNL Verbalizers

Once a user has created ontology elements, we need to consider how ontological information is presented in the context of a specific wiki page. There are clearly a number of options here, but, in accord with the general aim of improving end-user comprehension of ontological content, we present a CNL verbalization solution whereby users can view the logical statements associated with an ontology element using the Rabbit CNL.

The verbalization solution we present is based on the use of semantic templates, and all the usual advantages of wiki template solutions apply here. Thus, the CNL verbalization solution can be enabled in any existing or new SMW installation simply by copying the appropriate templates. Moreover, the end-user community can edit the templates in any way they see fit, perhaps to address template errors or to update specific templates in accord with the grammatical changes associated with particular CNLs. Finally, it is possible for entirely new templates to be created in order to reflect the introduction of a new CNL specification. The wiki environment supports the collaborative editing of CNL verbalization templates in the same way that wikis typically support content editing (i.e. modification of the source page associated with the wiki article). Moreover, users can exploit the discussion features of the MediaWiki system to initiate a discussion thread regarding a specific CNL verbalization template. This enables the template construction/modification effort to benefit from the collective knowledge of a potentially diverse user community (for instance, domain experts, linguists, casual end-users and wiki specialists may all contribute to a discussion about how a specific CNL verbalization component should be adapted to meet a particular requirement).

All the Rabbit sentences for a particular ontology are rendered using a top-level CNL verbalization template, called “Template:CNL.Rabbit”. This template calls on other templates to support the sentential serialization of ontological content to the Rabbit CNL. All the Rabbit CNL templates are stored in the “Template:” namespace, and specific templates are responsible for the generation of particular kinds of sentence. Thus, consider the Rabbit sentence: “Every Pashtun Tribal Chief is a kind of Tribal Chief”. This sentence is generated by a specific template, namely the “Template:CNL.Rabbit.getConceptRelationAssertions” template. Part of the mark-up associated with this template is listed below:

```
{{#vardefine:label|{{CNL.getLabel|{{{1}}}} }} }}{{#vardefine:super
| {{#ask: [[:{{{1}}|{{FULLPAGENAME}}}}}]] |?Category= |mainlabel=-|
format=list|link=none}} }} {{#if: {{#var:super}} |{{#arraymap:
{{#var:super}}|,|xxx|<li>Every [[:{{{1}}}}|{{!}}|{{#var:label}}]] is
a kind of [[:xxx|{{CNL.getLabel|xxx}}]]|{{{2}}}} }}|}}
```

This template works by retrieving the label associated with the subject of the sentence using the Template:CNL.getLabel” template. This label (“Pashtun Tribal Chief”) is assigned to the variable ‘label’ and the value of the variable is subsequently used in the phrase “Every <label> is a kind of” (note that Rabbit function words are highlighted in bold text in the above mark-up). A similar method is used for retrieving the label associated with the superclass of the subject class (Category:TribalChief in the current example) and appending it to the Rabbit sentence.

Most templates need to retrieve information from the SMW knowledge base in order to substitute values into Rabbit sentences. This is accomplished using the SMW ‘#ask’ parser function, which executes a query against the SMW knowledge base. Because the structure of the ontology is stored in the knowledge base, the ‘#ask’ parser function can retrieve information that was previously created using the form-based interface described in Section 4.3. In addition, because the Rabbit verbalization templates are not sensitive to the mechanism used for asserting ontological information, they can also exploit the semantic annotations that are associated with content represented elsewhere on the wiki page. If, for example, we had a page about Hamid Karzai in the wiki (i.e. a page with Hamid Karzai as the subject of the page) and included the following wiki-text on the wiki page’s source page, then the Rabbit CNL verbalization templates would generate the sentence “Hamid Karzai is president of Afghanistan”.

```
As of July 2009, Hamid Karzai is president of
[[IsPresidentOf::Afghanistan]].
```

At present, all Rabbit verbalization templates are embedded in the top-level SMW-mOWL templates for particular ontology elements (for example, all the Rabbit verbalization templates for named OWL classes are embedded in the SMW-mOWL “Template:NamedClass” template). In future work, we aim to make the use of Rabbit CNL verbalization capabilities more flexible by creating a separate “Template:RabbitBox” template that will render Rabbit sentences wherever it is embedded on a wiki page.

5 Problems and Issues

As part of our work on the WikiOnt-CNL system, we have implemented a number of ontologies to test the ontology authoring and CNL verbalization capabilities described herein. These efforts have revealed a number of problems and issues that we hope to address in future work. The current section provides an overview of some of the more important issues.

- **Embedded linguistic knowledge and CNL rendering.** In English, as with most NLs, a word can take on various forms when used in a particular sentence. To borrow an example from Kaljurand [18], the transitive verb ‘to border’ can be used in

three forms: “Every country that *borders* more than 2 countries that *border* a country that is *bordered* by...”. The morphological realization of these surface forms depends on access to linguistic knowledge about the base lexical form of an ontology element (e.g. “wiki:isLocatedAt” has the lexical form “is located at”), as well as knowledge about how inflected variants of the lexical form should be used in different grammatical contexts. For example, if a property is represented by a lexical form that is a transitive verb (e.g. “Property:Contains”), then it is important that the third person singular form of the verb (i.e. “contains”) is used in sentences featuring a singular subject, while the third person plural of the verb (i.e. “contain”) is used with plural subjects. This basic form of number agreement is relatively straightforward to implement in the case of transitive verbs, but not all properties have lexical counterparts that are transitive verbs. Kuhn [21], for example, identifies four types of property based on their linguistic characteristics: transitive verbs (e.g. “constrains”), transitive verb/adverb combinations (e.g. “quickly constrains”), ‘of’ constructs (e.g. “part of”), and adjectives in the comparative form (e.g. “larger than”). The correct rendering of these various properties in different CNL sentences is not something that can be accomplished without a fair amount of background linguistic knowledge, so it seems as though the ontology authoring capabilities in WikiOnt-CNL need to be extended to include the representation of linguistic knowledge. CNL-based ontology authoring tools already include support for the representation of this information. Thus, AceWiki allows users to create properties based on their linguistic category, e.g. transitive verb, transitive adjective, and so on. Furthermore, in the case of transitive verbs, the user can add additional linguistic knowledge to the property specification (i.e. they can specify the third person singular, plural and past participle forms of the verb). Of course, CNL-based ontology editors are at an advantage here because their interface is already set-up to support the entry of linguistically-oriented input. In AceWiki, for example, users indirectly create ontology elements by specifying words within given linguistic categories.

- **Resolving residual comprehension problems.** Despite being presented in NL, the semantics of some CNL sentences can still be difficult for human end-users to understand. In Rabbit, for example, we encounter the problem of specific sentences (e.g. “Relationship has part is transitive.”) being hard to understand for non-ontologists [8], while in ACE we see evidence for the usual confusion between the precise meanings of ‘and’ and ‘or’ expressions [17]. As long as the empirical evidence indicates the possibility for ambiguity or confusion with CNL sentences, it is important that we take steps to minimize such confusion. One potential solution to this problem is demonstrated by AceWiki. AceWiki enables content editors to add comments to specific ACE sentences using plain (unconstrained) NL (see [22]). Such comments can clarify the meaning of specific sentences in cases where comprehension problems might arise, and they therefore serve as an additional comprehension aid to the human end-user. Unfortunately, the implementation of such a solution in the case of the WikiOnt-CNL system is not straightforward. This is because CNL sentences are not explicitly represented in the wiki database; instead, they are dynamically generated each time a wiki page is loaded. One potential solution strategy here would be to exploit the research outcomes of CNL user evaluation studies in order to identify the kinds of sentences that pose potential

comprehension problems for human end-users. With this understanding in place, it might be possible to automatically add comments to certain types of sentence as and when they are rendered by the wiki system. For example, if we know that users have difficulty comprehending the meaning of a Rabbit sentence such as “Relationship has part is transitive.”, then we could automatically create a comment for this sentence that reads as follows: “For example, if objectX has part objectY and objectY has part objectZ then objectX also has part objectZ”. The mechanism for including these clarification comments would rely on the same template-mediated text substitution mechanism that is applied in the case of plain CNL sentences. The difference, in this case, would be the programmatic identification of potential problem sentences and the subsequent generation of comments that have been *empirically demonstrated* to improve end-user comprehension. Obviously, the implementation of this capability requires further research on the both the CNL user evaluation and technology development fronts¹⁰.

- **Rules and reasoning.** SMW has very limited support for reasoning. This is partly a result of the fact that SMW does not avail itself of particularly expressive knowledge representation capabilities (in addition the creators of the SMW system wanted to keep performance overheads to a minimum (see [6]). With the addition of more expressive modeling capabilities within WikiOnt-CNL, it becomes possible to implement more sophisticated forms of reasoning, and, in some cases, the results of such reasoning could be included in a page and rendered as Rabbit sentences. An initial attempt at implementing reasoning capabilities within SWM is proposed by Bao et al [20]. Bao et al [20] suggest that a combination of inline queries, in conjunction with the template mechanism described here, could be used to support certain forms of rule-based processing with little or no modification of the underlying SMW software. Given that the results of such reasoning processes can be represented as RDF triples, it seems plausible that a future extended version of the WikiOnt-CNL system could serialize the results of reasoning processes as CNL sentences and embed them in specific wiki pages.
- **Ontology authoring capabilities.** Finally, despite the existence of a form-based editing interface, it is still rather awkward to create ontologies in WikiOnt-CNL. The main problem here relates to the fact that complex class expressions need to be encoded as relationships between multiple pages (often corresponding to anonymous classes). In order to address this problem, it will be important to consider the development of integrated CNL (and perhaps advanced graphical) editing interfaces. These interfaces could support the visualization and editing of entire ontologies without requiring the user to interact with multiple wiki pages.

¹⁰ An alternative, and somewhat more straightforward, solution involves the inclusion of a hyperlink to a separate wiki page (a “CNL Sentence Page”) for each CNL sentence that is generated within the wiki. Users could use the hyperlink to access the CNL Sentence Page and add sentence-specific content to the page using standard wiki editing techniques. Because the name of the CNL Sentence Page would be the same as the CNL sentence it represents, the CNL Sentence Page could be used to represent the same sentence irrespective of where it was rendered in the wiki. Moreover, in the manner typical of wiki systems, the page would only be generated when the user actually created some content, and a visual indicator (e.g. font colour change) could be used to indicate whether a specific CNL sentence was associated with user-generated content.

6 Future Work

The following are focus areas for future work on the WikiOnt-CNL system.

- **Analysis and incorporation of linguistic knowledge.** In the context of future work related to the WikiOnt-CNL system, we plan to undertake a comprehensive survey of the different kinds of properties used in SW ontologies. This analysis will help us to understand the linguistic knowledge required for an appropriate serialization of ontology properties in grammatically-diverse sentential contexts. The results of this work will guide the development of extensions to both the SMW-mOWL and the form-based ontology editing interface.
- **Implementation and testing of Special Page capabilities.** As discussed in Section 4.1 some of the functionality of the WikiOnt-CNL is realized by Special Page components. Some of these components require additional work before they can be fully exploited by users of the system (e.g. the CNL Parser Special Page is currently incomplete).
- **Development of additional CNL verbalizers.** WikiOnt-CNL currently provides verbalization templates for the Rabbit CNL, but we are also planning to develop templates for SOS and ACE. While we recognize the efforts currently underway to develop a standard CNL for OWL ontologies¹¹, we suspect that different user communities may still want to exploit one of the extant CNLs. It may be the case, for example, that CNLs are differentially suited to certain editing or knowledge communication functions. If so, then it makes sense for collaborative knowledge editing environments to accommodate multiple CNLs and be largely agnostic about the kind of CNL interfaces they make available to end-users.
- **Development of a CNL-editing interface.** As discussed in Section 5, it is important to consider the addition of CNL-editing interfaces to the WikiOnt-CNL system. Ideally, this editing capability should be available to users in addition to (and not instead of) the other ways of interacting with semantically-enriched content (see Section 3).

7 Conclusion

The SW provides a vision of the future potential of the World Wide Web to support advanced forms of information processing and Web-enabled intelligence. In this paper, we have discussed our efforts to contribute to the realization of this vision. We have suggested that a combination of both semantic wikis and CNLs may provide the right mix of features necessary for large numbers of users to become more involved in the creation (and exploitation) of online semantic content. The focus of our efforts to date has been on the highly popular semantic wiki system, SMW, as well as the Rabbit CNL. We have shown that by using a template-based mechanism, we can successfully support the representation of highly expressive ontologies within SMW, with no changes to the underlying software. Furthermore, we have shown that it is possible to use semantic templates to support the serialization of semantic content (including standard SMW page annotations) to Rabbit sentences. The particular advantages of

¹¹ See <http://code.google.com/p/owl1-1/wiki/OwlCnl>

this solution strategy include the open nature of the template-based mechanism; i.e. the CNL verbalization templates can be collaboratively edited in the same kind of way as any other wiki page. Members of the end-user community can use the collaboration features of the host wiki system to adapt CNL verbalization capabilities to suit community-specific requirements. They can also extend the CNL verbalization capabilities so as to support the generation of alternative CNLs.

Our future work on the WikiOnt-CNL system seeks to enhance its current functionality. In addition to the implementation of an embedded CNL editor, we propose to extend the verbalization capabilities by using embedded linguistic knowledge. Such features will, we suggest, provide us with a technological platform from which to assess the effect of user-friendly collaborative knowledge editing systems on the general availability of high-quality semantically-enriched content.

Acknowledgments. This research was sponsored by the US Army Research laboratory and the UK Ministry of Defence and was accomplished under Agreement Number W911NF-06-3-0001. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the US Army Research Laboratory, the U.S. Government, the UK Ministry of Defence, or the UK Government. The US and UK Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

References

1. Horridge, M., Drummond, N., Goodwin, J., Rector, A., Stevens, R., Wang, H.H.: The Manchester OWL Syntax. In: OWL Experiences and Directions Workshop (OWLED 2006) at ISWC 2006, Athens, Georgia, USA (2006)
2. Rector, A., Drummond, N., Horridge, M., Rogers, J., Knublauch, H., Stevens, R., Wang, H., Wroe, C.: OWL Pizzas: Practical Experience of Teaching OWL-DL: Common Errors & Common Patterns. In: Motta, E., Shadbolt, N.R., Stutt, A., Gibbins, N. (eds.) EKAW 2004. LNCS (LNAI), vol. 3257, pp. 63–81. Springer, Heidelberg (2004)
3. Hewlett, D., Kalyanpur, A., Kovlovski, V., Halaschek-Wiener, C.: Effective Natural Language Paraphrasing of Ontologies on the Semantic Web. In: End User Semantic Web Interaction Workshop, Galway, Ireland (2005)
4. Smart, P.R.: Controlled Natural Languages and the Semantic Web, Technical Report, School of Electronics and Computer Science, University of Southampton, Southampton, England (2008)
5. Smart, P.R., Engelbrecht, P.C.: An Analysis of the Origins of Ontology Mismatches on the Semantic Web. In: Gangemi, A., Euzenat, J. (eds.) EKAW 2008. LNCS (LNAI), vol. 5268, pp. 120–135. Springer, Heidelberg (2008)
6. Krötzsch, M., Vrandečić, D., Völkel, M., Haller, H., Studer, R.: Semantic Wikipedia. *Journal of Web Semantics* 5, 251–261 (2007)
7. Dolbear, C., Hart, G., Kovacs, K., Goodwin, J., Zhou, S.: The Rabbit Language: Description, Syntax and Conversion to OWL, Technical Report, Ordnance Survey, Southampton, UK (2007)

8. Hart, G., Johnson, M., Dolbear, C.: Rabbit: Developing a Control Natural Language for Authoring Ontologies. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) ESWC 2008. LNCS, vol. 5021, pp. 348–360. Springer, Heidelberg (2008)
9. Kuhn, T.: AceWiki: A Natural and Expressive Semantic Wiki. In: Semantic Web User Interaction Workshop at CHI 2008, Florence, Italy (2008)
10. Kuhn, T.: AceWiki: Collaborative Ontology Management in Controlled Natural Language. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) ESWC 2008. LNCS, vol. 5021, Springer, Heidelberg (2008)
11. Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. *Scientific American* 284, 34–43 (2001)
12. Giles, J.: Internet encyclopaedias go head to head. *Nature* 438, 900–901 (2005)
13. Hepp, M., Bachlechner, D., Siorpaes, K.: OntoWiki: community-driven ontology engineering and ontology usage based on Wikis. In: International Symposium on Wikis, Odense, Denmark (2006)
14. Cregan, A., Schwitter, R., Meyer, T.: Sydney OWL Syntax—towards a Controlled Natural Language Syntax for OWL 1.1. In: OWL Experiences and Directions Workshop, Innsbruck, Austria (2007)
15. Kaljurand, K., Fuchs, N.E.: Bidirectional mapping between OWL DL and Attempto Controlled English. In: 4th Workshop on Principles and Practice of Semantic Web Reasoning, Budva, Montenegro (2006)
16. Engelbrecht, P.C., Hart, G., Dolbear, C.: Talking Rabbit: a User Evaluation of Sentence Production. In: Workshop on Controlled Natural Language (CNL 2009), Marettimo Island, Italy (2009)
17. Kuhn, T.: How to Evaluate Controlled Natural Languages. In: Workshop on Controlled Natural Language (CNL 2009), Marettimo Island, Italy (2009)
18. Kaljurand, K.: ACE View - An ontology and rule editor based on Attempto Controlled English. In: 5th OWL Experiences and Directions Workshop (OWLED 2009), Karlsruhe, Germany (2008)
19. Denaux, R., Dimitrova, V., Cohn, A., Dolbear, C., Hart, G.: Rabbit to OWL: Ontology Authoring with a CNL-based Tool. In: Workshop on Controlled Natural Language (CNL 2009), Marettimo Island, Italy (2009)
20. Bao, J., Ding, L., Smart, P.R., Braines, D., Jones, G.: Rule Modeling Using Semantic MediaWiki. In: 3rd Annual Conference of the International Technology Alliance (ACITA 2009), Maryland, USA (2009)
21. Kuhn, T.: Attempto Controlled English as an Ontology Language. In: Reasoning on the Web with Rules and Semantics, Munich, Germany (2006)
22. Kuhn, T.: How Controlled English can Improve Semantic Wikis. In: 4th Workshop on Semantic Wikis, Heraklion, Greece (2009)

A Controlled Language for the Specification of Contracts

Gordon J. Pace¹ and Michael Rosner²

¹ Dept. Computer Science
University of Malta, Msida MSD2080, Malta

² Dept. Intelligent Computer Systems
University of Malta, Msida MSD2080, Malta
{gordon.pace,mike.rosner}@um.edu.mt

Abstract. Controlled natural languages have been used to enable the direct translation from natural language specifications into a formal description. In this paper we make a case for such an approach to write contracts, and translating into a temporal deontic logic. Combining both temporal behaviour and deontic behaviour is challenging both from a natural language and a formal logic perspective. We present both a logic and a controlled natural language and outline how the two can be linked.

1 Introduction

Legal contracts are useful artifacts which have evolved to regulate behaviours in agreed ways. An important feature of a contract is that one party makes an offer for an arrangement that another accepts. Once this exchange takes place, constraints are enforced over the parties' future actions. If these constraints are broken, a *breach of contract* is recognised and remedies may be provided. Contracts can operate with varying degrees of formality and complexity. Much of the time, contracts are made orally or are implied by the situation at hand. However, beyond a certain level of complexity, written contracts are the norm. These are typically expressed in natural language without any particular restrictions on the form of language. The use of unrestricted natural language for the formulation of contracts is something of a double-edged sword.

On the one hand, it allows quite complex specifications of contractual properties to be formulated concisely. For example, the clause¹ “*We shall not be held liable for any damages resulting from the disruption or malfunction of the DNS service.*” seems pretty clear, relieving one party from the specific obligation to pay damages resulting from a certain class of events. Most of the complexity follows from the meanings of words like “malfunction”, and boils down to a three-place relation of the form “X shall not be held liable for Y resulting from Z” with appropriate constraints on the variables.

On the other hand, unrestricted natural language brings with it a host of well-known problems relating to ambiguity, syntactic complexity, context sensitivity, etc.

¹ Extracted from the Terms and Conditions of NIC(Malta), a local domain name registration service.

For example, a certain contract specifies that “*the payment adjustment per tonne will apply to the quantity of asphalt cement in the hot mix accepted into the work during the month for which it is established*”. Amongst the problems are (i) identification of the referent of the word “it” (which could be almost any of the preceding noun phrases), (ii) deciding whether “during . . . established” modifies the application of the payment adjustment or the quantity of asphalt cement in the hot mix.

Common sense will sometimes tell us which of these interpretations is the correct one, but this is not always the case. Certainly important legal cases have been won or lost according to the way in which a scope ambiguity is resolved. Natural language contracts that specify obligations between contracting parties are so complex that we are generally forced to rely upon costly legal specialists for their formulation and for analysis of their implications.

These considerations motivate the idea of a *controlled language* for certain classes of contract. Controlled languages (cf. Pulman [1]) are “designer languages” in the sense of being artificially restricted subsets of natural language that have been designed with a mission in mind. The mission typically involves the facilitation of learning, translation, analysis, or some combination of these as in explanation. The specific restrictions are mission-dependent and might concern vocabulary, syntax and semantics.

In our case we are seeking a controlled language which is rich enough to express the key concepts, simple enough for ordinary people to use, and yet precise enough to have a semantics which is amenable to automated reasoning methods.

Automated reasoning could be the basis for a tantalising range of support tools providing help in the areas of contract formulation, explanation, and applicability with respect to a particular set of circumstances or scenario e.g. deducing whether a given action is obliged, prohibited, or merely licenced.

For the purpose of developing this idea, a great deal will turn on what we mean by a class of contracts and which class to choose. In this article, we focus on contracts which occur within fairly narrow computational settings, where the underlying reality is well enough understood and sufficiently precise to enable a range of contractual issues to be explored.

Contracts appear in computational settings in various guises — from simple pre- and post-condition pairs in programming giving guarantees on outcomes based on assumptions on the initial conditions, to quality of service agreements requesting, for instance, a maximum guaranteed packet-dropping probability. Service-level agreements, contracts used for conformance checking, and system specifications are expressed in a variety of ways, but may still contain interesting deontic nuances. Simple examples of a system specification, combining deontic and temporal notions are given by the following sentences:

- Upon accepting a job, the system guarantees that the results will be available within an hour unless cancelled in the meantime.
- Only the owner of a job has permission to cancel the job.
- The system is forbidden from producing a result if it has been cancelled by the owner.

One major challenge in the use of controlled natural languages (CNLs) is the choice of both the natural sublanguage itself and the logic used to capture its meaning. This

choice is crucial to enable us to go back and forth between the two levels when parsing, reasoning about or manipulating objects in the domain, and generating of natural language explaining the results.

In this paper, we present the use of CNL techniques for the specification of contracts. We identify an appropriate logic and controlled language to enable the automated analysis of CNL contracts in a formal and precise manner. One important choice which we discuss at length is in the choice of a complementary pair of a controlled language and logic, so as to enable reasoning and manipulation on either side to be transferable to the other. However, the choice of a compatible logic and language pair still leaves much choice as to the structural abstraction possible in the representations.

One contribution of the paper is a discussion on further choices to make to approach the ideal level of abstraction. Since manipulation and analysis at the logical level may result in changing the structure of the original body of text, referring back to the original content can be challenging. Even keeping cross references between the two entities may not be simple — and we address the issue by trying to keep a one-to-one relationship between the two domains at the syntactic level. One technique to enable easier automated manipulation of the contracts keeping the syntactic structure intact is one developed by the programming language community — using embedded languages. The idea behind the approach is to enable expressions written in a domain specific language to be written as though they were part of the code written in the host language in which one is programming. This allows terms of the logic not only to appear in the host language — but also to be manipulated as data objects. In this way as much as possible of the syntactic structure of the original information is retained in the reasoning at the logical level.

The paper is structured as follows. In Section 2 we discuss the issues arising in choosing the right controlled natural language and logic pair. We then present the deontic logic in Section 3, and show how it can be embedded in Haskell and what benefits are gained in Section 4. In Section 5, we present the controlled natural language and its transformation into the logic. We finally conclude in Section 7.

2 Controlled Languages and Translation

The end-user system that we are aiming for can be regarded as an expert-system specialised in the domain of computer-oriented contracts. The core expertise of the system resides in its ability to reason logically about the properties of contracts themselves (e.g. concerning a contract's internal consistency) and about the status of actions or events falling within the scope of the contract - for example, whether a certain action is prohibited or obligatory. What makes this system different is that communication with it takes place in natural language. We have already hinted at different kinds of communication task that such a system might undertake, including authoring, explanation, question answering, what-if style querying etc. Underlying these tasks is an object language for the specification of contracts. This underlying language is the CNL that concerns us here.

The idea, then, is that the system can effect two-way communication in CNL. In order to reason about the properties of a contract expressed in CNL, they system must translate it into a representational form which is suitable for performing inferences. To

communicate the results of those inferences back to the user, there must be the means to translate relevant elements of the (possibly modified) representational form back into CNL.

Much of the discussion that follows concerns an appropriate choice of logic representation and the level of abstraction that it encodes. The discussion recalls issues that arise with respect to the choice of intermediate levels of representation needed for transfer-based Machine Translation. This is depicted in Figure 1 (a) below which depicts so-called Vauquois triangle (Vauquois [2]).

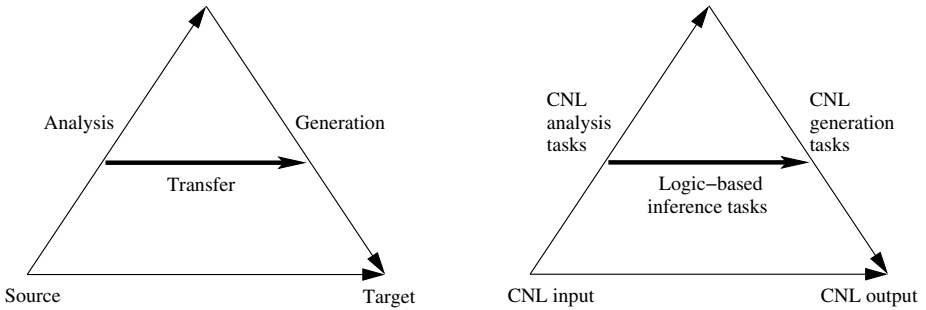


Fig. 1. (a) Vauquois Triangle for machine translation; and (b) Controlled natural language triangle

The bottom left and right vertices of the triangle represent source and target language sentences respectively. In a classic transfer-based system, the translation process comprises three phases: (i) analysis (ii) transfer and (iii) generation. The actual translation is carried out by the transfer phase which applies *transfer rules* to abstract *source sentence representations* to yield *target-language representations*. The great advantage of defining translation over representations rather than sentences or sentence-fragments is that transfer rules are able to translate *classes* of similar linguistic forms and hence to capture linguistic generalisations. The downside is that representations are not sentences, and in order to ground the system to concrete text, appropriate mappings to source and target text have to be defined. These are supplied by the analysis and generation phases, which are respectively responsible for the mapping between source text and representation, and between target representation and target text.

There is a close relationship between transfer-based machine translation and the CNL inference system we are trying to construct. This is illustrated in figure 1(b), where the “source text” is in CNL and expresses at least (i) a contract and also (ii) some other information concerning the particular task at hand, for example concerning the contract’s consistency, request for explanation, or applicability of the contract to a particular situation. In this article we will limit the discussion to (i). The right hand “target” vertex of the triangle represents CNL output, i.e. a piece of text that fulfils the task at hand. In between, corresponding to the place where transfer is carried out in the machine translation system, is where the relationship between the task, and the answer, is computed using logical inference. So the representation, whatever it is, encodes expressions

in the logic, and logical inference maps between sets of logic expressions encoded as representations. And just as before we had analysis and generation to connect these representations to the concrete world of sentences, so we now require analogous, if rather different processes, to analyse CNL into logic and vice versa.

The geometrical dimensions of the Vauquois triangle provide some insight into the tradeoffs at play between deep and shallow levels of linguistic representation, as shown in figure 2. A shallow level of representation, depicted by the dotted line, is close to the surface text. Accordingly, analysis and generation are relatively simple, since they only need traverse a short (vertical distance). At the same time, the transfer operation is large - since it has to cope with a low level of abstraction, the similarity between similar cases is lost.

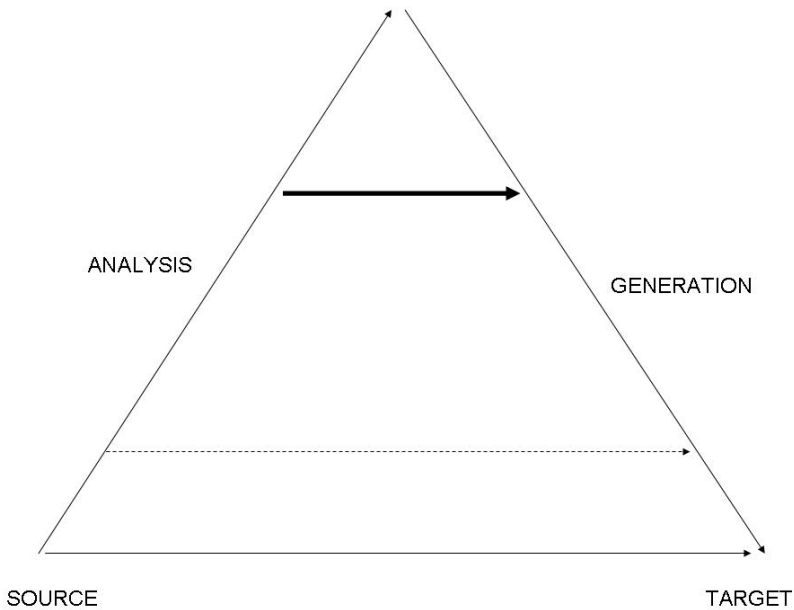


Fig. 2. Deep or shallow linguistic representations

Conversely deep level transfer, represented by the upper, thick solid line, is short, meaning that transfer is relatively simple. This is because it is carried out at a high level of generalisation. But the journey to reach that level is long: analysis and generation are both complex.

So from a machine translation perspective, the choice of representation level is by no means straightforward. In arriving at an appropriate choice, a number of different considerations must be borne in mind including in particular the language pairs under consideration, the extent to which the complexity of monolingual processes (analysis and generation) is to be traded off against the bilingual one (transfer).

From the perspective of inference, the issue is the choice of logic to use. Here the primary requirement is that it has to be sufficiently expressive to encompass the domain of the controlled language. However, this leaves a wide spectrum of possibilities of the abstraction level of the logic as shown in figure 3.

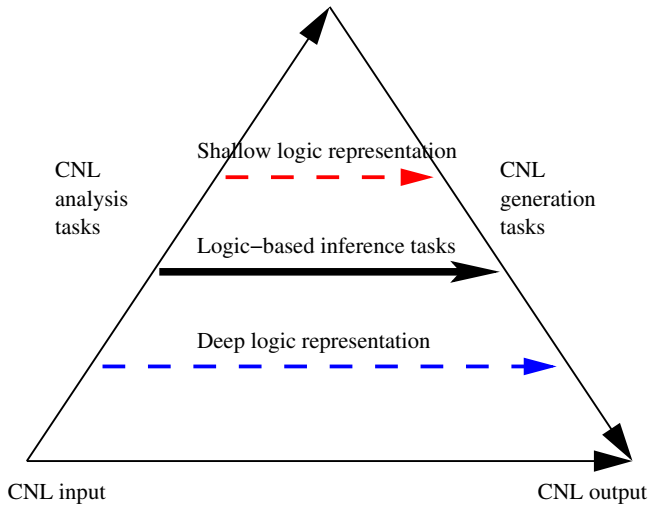


Fig. 3. Deep or shallow logical representations

At one extreme, one can choose a logic with a minimal set of orthogonal operators with no overlap in semantic expressiveness; or, at the other extreme, opt for a logic having a rich set of operators with overlapping semantics, but which may allow more concise descriptions in different scenarios. This is usually considered to be a language design choice and challenge, which is faced whenever a new language is designed — e.g. should we have both a `repeat...until` and a `while...do` construct in the same language despite the fact that they are expressible in terms of each other? When designing a programming language one regularly opts for redundancy in the operators, if it aids the description of different algorithms. However, when building a logic, the choice is usually that of limiting the number of operators to simplify giving a semantics to the logic, and to ensure soundness. Other redundant, or higher-level constructs, are simply seen as syntactic-sugar, reducible to the underlying, syntactically-miserly logic.

Going for a low-level logic with few operators, but complete with respect to the domain of application, is the choice typically made by logicians when designing a new logic, and has various advantages, the biggest of which is that the axiomatisation is limited to just a few, simple operators. Furthermore, analysis from language to logic, and transformations within the logic are typically straightforward to define and prove correct. However, this comes at the price of a loss of information implicitly available in the structure had one to use other operators.

Consider propositional logic, in which all operators can be expressed in terms of the nand (\uparrow) operator. However, expression the implication $p \rightarrow q$ using the nand operator results in $p \uparrow (q \uparrow q)$ giving no clue of the original intended term. If the controlled natural language caters for implication, there is the need for a closely corresponding operator in the logic if natural language generation is to be performed in a reasonable manner.

The alternative approach, in which we add various new operators, to have a closer correspondence between the controlled natural language and the logic adds work for the logician, who must ensure that the operators are sound with respect to each other, that their definitions introduce no contradictions and satisfy algebraic laws which the logic is expected to obey.

The solution we adopt in this paper is to go for a logic with few operators (which do however correspond, as closely as possible, to the CNL constructs). As usual, for higher abstraction layers, other operators not included in the basic logic are introduced as syntactic sugar. However, in reasoning about the logic, manipulation of expressions and generation of formulae, we try to retain the structure introduced through the abstraction layers in the logic expressions. For instance, in the propositional logic given earlier, although we may choose to include only nand in the basic logic, and introduce implication as syntactic sugar. However, when storing expressions involving implications, we try to maintain this information throughout the reasoning process, allowing generation of natural language sentences which use implications and not just nand. The reduction to the basic operators is only done when reasoning process requires it. This approach enables the use of a simple and elegant logic, but still enables reasonable generation of natural language. This results in a two-level approach with high-level operators which are only opened up when required.

3 Underlying Logic Representation Language

It is crucial to identify a sufficiently expressive logic to enable the description of and reasoning about contracts. In computer science, contracts have frequently been expressed as properties which should be satisfied by the system bound by the contract. This view enables a straightforward characterisation of contracts, but does not enable (i) reasoning *about* the contract — asking questions such as ‘What are the currently undischarged obligations in the contract?’; or (ii) reasoning about exceptional cases in a contract — such as ‘whenever clause (a) is violated, the user is prohibited from obtaining the service’. The introduction of explicit prohibition, obligation and permission clauses into the contract is thus essential to enable reasoning about contracts.

An important choice is between adopting an action-based or a state-based deontic logic. In the former, the deontic operators function over actions — one is obliged to deliver a service, as opposed to the latter, in which the operators function over the state, one is prohibited from reaching a state in which the balance is negative. The choice depends mainly on the type of contracts we choose to translate into logic. We choose to adopt the action-based approach.

Contracts constrain the behaviour between different agents, and it is thus important to include information about these participating agents. We choose to tag deontic expressions with an actor to identify which entity is being constrained by the clause.

Contracts are obviously dependent on the notion of choices — eg ‘if the client connects, then the client is obliged to pay’. Regularly, contracts go further, and make choices based on the enacted clauses in the contract itself eg ‘as long as the client is obliged to pay, the client is forbidden from closing the account’. Clearly, this introduces a notion of self reference, which can be dangerous — eg ‘if the client is forbidden from closing the account, then the client is permitted to close the account’, which would require the analysis of self-referential clauses similar to the ones used in the analysis of hardware with combinational cycles.

The challenge to formalise deontic logic, to reason about normative concepts such as obligations and permissions, has been a hot topic of research for several decades. The main challenge is that it is very easy to describe paradoxical situations using deontic concepts [3]. Paradoxes typically follow from laws which one expects to hold of the deontic operators. For instance, one may consider it natural that obligations should be monotonic. After all, if you are obliged to do something, then you are also obliged to do something weaker (and more): if $p \rightarrow q$, then $O(p) \rightarrow O(q)$ — where $O(p)$ means that there is the obligation to perform p or reach a state satisfying it) However, a naïve introduction of this law results in Ross’s Paradox: from the statements ‘You are obliged to post the letter’, one can infer that ‘You are obliged to post the letter or burn it’, which is seemingly satisfiable by burning the letter. Without resorting to mutually exclusive actions and states, this interpretation is clearly counterintuitive. Similarly, using monotonicity and allowing reasoning about the state of affairs both inside and outside the deontic operators, leads to the Good Samaritan Paradox: from the statement that ‘It ought to be the case that Jones helps Smith who has been robbed’ we would be able to conclude that ‘It ought to be the case that Smith is robbed,’ which is also not expected.

Introducing the concept of time brings further paradoxes [4]. Consider the law of a country which says that: ‘You are obliged to hand in Form A on Monday and Form B on Tuesday, unless officials stop you from doing so.’ On Monday, John spent a day on the beach, thus not handing in Form A. On Tuesday at 00:00 he was arrested, and brought to justice on Wednesday. The police argue: ‘To satisfy his obligation the defendant had to hand in Form A on Monday, which he did not. Hence he should be found guilty.’ But John’s lawyer argues back: ‘But to satisfy the obligation the defendant had to hand in Form B on Tuesday, which he was stopped from doing by officials. He is hence innocent.’ Both interpretations seem intuitively acceptable — but are clearly incompatible since they use different interpretations of the moment of violation of *a sequence* of obligations.

Various axiomatizations have been proposed, in an attempt to deal with the paradoxes. However, one of the more effective approaches has been that of restricting the syntax [5]. Since our aim is to reason about contracts derived from natural language texts, and which could thus include paradoxes or contradictions, we opt for a more general logic, which could then be restricted, syntactically or semantically, to weed out potential problems.

The logic we adopt will have operators closely matching those found to be used in the CNL, but expressing some of them using syntactic sugar so to avoid redundancy. The deontic logic we build has three basic underlying deontic concepts: prohibition, permission and obligation. All three of these will always appear tagged with an agent specifying to whom the deontic operator applies. Since we would want our contracts to be phrased in terms of actions — for example, which ones are permitted at a particular point in time, we adopt an action-based deontic logic. The underlying actions can occur concurrently, and we will use the notation \overline{act} for any action different from action act . Action expressions, are defined in terms of regular expression operators, with conjunction. In addition to basic actions, one can also check whether an obligation, permission or prohibition is currently enacted: $O?(A.\alpha)$ is satisfied if agent A is obliged to perform α at this moment in time, similarly $P?(A.\alpha)$ and $F?(A.\alpha)$ are used to check for permission and forbidden actions.

$$\begin{aligned}
 \text{action-expression} ::= & \text{basic-action} \mid \overline{\text{basic-action}} \\
 & \mid O?(agent.\text{action-expression}) \\
 & \mid P?(agent.\text{action-expression}) \\
 & \mid F?(agent.\text{action-expression}) \\
 & \mid \text{action-expression} \cdot \text{action-expression} \\
 & \mid \text{action-expression} + \text{action-expression} \\
 & \mid \text{action-expression} \& \text{action-expression} \\
 & \mid \text{action-expression}^*
 \end{aligned}$$

We will use lower case letters (a, b, c) for basic actions, Greek letters for action expressions (α, β, γ) and upper case letters (A, B, C) for agents.

The deontic logic includes obligation, permission and prohibition ($O(A : \alpha)$, $P(A : \alpha)$ and $F(A : \alpha)$), non-deterministic choice ($+$), conjunction ($\&$), conditional ($c_1 \triangleleft \alpha \triangleright c_2$), generalised sequential composition ($c_2 \blacktriangleleft c \blacktriangleright c_1$, which starts with c , and then follows it up with c_1 or c_2 depending on whether it was satisfied or violated) and timing information ($c_{[b,e]}$):

$$\begin{aligned}
 \text{contract} ::= & \top_{time} \mid \perp_{time} \mid O(agent : action) \mid P(agent : action) \mid F(agent : action) \\
 & \mid \text{contract} + \text{contract} \mid \text{contract} \& \text{contract} \mid \text{contract} \triangleleft action \triangleright \text{contract} \\
 & \mid \text{contract} \blacktriangleleft \text{contract} \blacktriangleright \text{contract} \mid \text{contract}_{[time,time]}
 \end{aligned}$$

Note that choice may appear both inside and outside a deontic operator e.g. $O(A : a + b)$ and $O(A : a) + O(A : b)$. These are semantically different: the contract $O(A : a + b)$, allows agent A to choose whether or not to perform a or b — either of the two would satisfy the contract, while the contract $O(A : a) + O(A : b)$ would be non-deterministic.

Using these operators and fix-point definitions, other operators can be defined: (i) one branch conditional: $e \rightarrow c \equiv c \triangleleft e \triangleright \top_0$; (ii) sequential composition: $c_1; c_2 \equiv c_2 \blacktriangleleft c_1 \blacktriangleright \perp_0$; (iii) the always operator: $\Box(c) \equiv c \& \top_1; \Box(c)$; and (iv) the sometimes operator: $\Diamond(c) \equiv c + \top_1; \Diamond(c)$. Furthermore, the complement of an action or an agent

can be expressed in the logic using a bar over the object. The examples given earlier can be written in the following manner:

- Upon accepting a job, the system guarantees that the results will be available within an hour unless cancelled in the meantime:
 $\Box(\text{accept}_j \rightarrow O(\text{system} : (\text{result}_j + \text{cancel}_j))_{[0,1hr]})$
- Only the owner of a job has permission to cancel the job:
 $\Box(P(\text{owner}_j : \text{cancel}_j) \ \& \ F(\overline{\text{owner}}_j : \text{cancel}_j))$
- The system is forbidden from producing a result if it has been cancelled by the owner: $\Box(\text{cancel}_j \rightarrow F(\text{system} : (\Diamond(\text{result}_j))))$

The logic proposed shares much in flavour with CL [5] and other action-based deontic logics. One can construct observer formulae (one for each actor), using which one can model-check contracts [6], and perform contract analysis. The temporal side of the logic is based on timed regular expressions [7]. Although, as in timed regular expression, a continuous time domain may be used, at the moment we restrict the time to a discrete domain for analysis techniques. Through the use of a trace semantics of the logic, standard model-checking techniques can be used to check for validity.

The deontic logic has been given an operational trace semantics, a portion of the untimed version of which can be seen in Figure 4. Enriched semantics with deontic information (what actions are permitted, prohibited and obliged at every point in time) and with information regarding the actor responsible for an action, and which actor violated or satisfied which contract can be similarly given. The extension of the semantics to deal with time uses the approach adopted in timed regular expressions [7], and adds a time-stamp to each action.

4 Embedding the Contract Logic in Haskell

When building tools to use and manipulate a logic or language, the approach is typically that of building and using a domain-specific library to deal with the concepts and ways of manipulating them. One alternative approach which is becoming more widespread is that of using embedded languages [8,9] — where one builds a domain-specific *language* within the host language, in which there is a direct correspondence to the syntax of the logic or language one is reasoning about. This approach enables the use of abstraction and modularization techniques from the host language in the embedded language. By building syntactic constructs and a type system of the domain-specific language within the host language, domain-specific programs can be expressed as objects in the host language. Apart from ways of representing syntactically the language in question, one also typically provides semantic interpretations of the domain-specific programs — enabling execution, visualization, analysis, etc. This approach has been used to model various domains, ranging from financial contracts [10] to hardware description [11].

The main difference between a domain-specific library in the general-purpose language and a domain-specific embedded language is that in the embedded language approach, careful design of domain-specific combinators creates the illusion that the domain-specific code fragments are ‘programs’ appearing as parts of the programs in

Obligation:

$$\frac{}{O(p : a) \xrightarrow{a} \top}$$

$$\frac{}{O(p : a) \xrightarrow{b} \perp}$$

Prohibition:

$$\frac{}{F(p : a) \xrightarrow{b} \top}$$

$$\frac{}{F(p : a) \xrightarrow{a} \perp}$$

Reparation:

$$\frac{}{c_1 \blacktriangleleft \top \blacktriangleright c_2 \longrightarrow c_1}$$

$$\frac{}{c_1 \blacktriangleleft \perp \blacktriangleright c_2 \longrightarrow c_2}$$

$$\frac{c \longrightarrow c'}{c_1 \blacktriangleleft c \blacktriangleright c_2 \longrightarrow c_1 \blacktriangleleft c' \blacktriangleright c_2}$$

$$\frac{c \xrightarrow{a} c'}{c_1 \blacktriangleleft \top \blacktriangleright c_2 \xrightarrow{a} c_1 \blacktriangleleft c' \blacktriangleright c_2}$$

Conjunction:

$$\frac{}{\perp \& c \longrightarrow \perp}$$

$$\frac{}{c \& \perp \longrightarrow \perp}$$

$$\frac{}{\top \& c \longrightarrow c}$$

$$\frac{}{c \& \top \longrightarrow c}$$

$$\frac{c_1 \longrightarrow c'_1}{c_1 \& c_2 \longrightarrow c'_1 \& c_2}$$

$$\frac{c_2 \longrightarrow c'_2}{c_1 \& c_2 \longrightarrow c_1 \& c'_2}$$

$$\frac{\begin{array}{c} c_1 \xrightarrow{a} c'_1 \\ c_2 \xrightarrow{a} c'_2 \end{array}}{c_1 \& c_2 \xrightarrow{a} c'_1 \& c'_2}$$

Choice:

$$\frac{}{c_1 + c_2 \longrightarrow c_1}$$

$$\frac{}{c_1 + c_2 \longrightarrow c_2}$$

Fig. 4. An excerpt of the formal semantics of the contract logic

the host language. By providing appropriate functions, the programmer may then generate, analyse and manipulate these ‘programs’ (written in the domain-specific language) as though they were part of the host language itself and thus, effectively make the host act as a meta-language for the embedded language. Clearly, the more flexible and high-level the host language and its syntax are, the more difficult it becomes to distinguish where the domain-specific program ends and where the rest of the code starts.

The structuring of embedded languages is ideal for the approach we require, in which we have access to the structure of the logic statements, manipulate them, opening up syntactic sugar only when and if required, and we have embedded part of the logic as an embedded-language in Haskell [12] — which has, through various case studies, been shown to be an excellent host language to provide the right modularity and abstraction to develop an embedded language.

Describing contracts: Although internally, the embedded-language uses an abstract datatype to keep information about the structure of the contract, the end user is provided with a set of functions and operators to describe contracts. For instance, the infix operators `<|` and `|>` correspond to the conditional operators \triangleleft and \triangleright , while the operators `<<` and `>>` correspond to the reparation operators \blacktriangleleft and \blacktriangleright . Deontic operators are similarly defined using the functions `permission`, `obligation` and `forbidden`, all of which take two operands — the agent and the action. Agents and actions are defined using the constructors `agent` and `action` which take a string.

As a simple example of writing a contract using the embedded language directly, we show below a simple contract which branches on whether the agent picks up an item in a shop. If the item is not picked up, then the contract terminates successfully, but if it is, an obligation to pay is enacted, which can either be satisfied, thus permitting the agent to leave the shop, or not, in which case an obligation to return the object on the shelf is enacted:

```
stroll :: Agent -> Contract
stroll a = shop a <| pickupItem |> success

shop :: Agent -> Contract
shop a =
  permission(a, leaveShop)
    << obligation(a, pay) >>
    obligation(a, returnObject)
```

Note that in our setting, most of this code would be generated directly from the CNL.

Families of contracts: One advantage of using the embedded language approach is that regular contracts can be described using higher-order contracts. This approach of writing parametrised contracts is particularly useful to enable syntactic sugar to be handled in a direct manner. Consider an operator to retry satisfying a contract a number of times. The combinator can be expressed in the embedded language in the following manner:

```

retry :: Integer -> Contract -> Contract
retry 1 c = c
retry n c = success << c >> retry (n-1) c

```

Contract transformations: The embedded language also enables the writing of transform contracts syntactically. Using functions to check the type of operator and its operands, users can write their own transformations of contracts. The following code extract shows how one can change a contract selectively, allowing a softer form of obligations for a particular action:

```

retryObl :: Action -> Contract -> Contract
retryObl a c
  | isConditional c =
    retryObl a (leftBranch c)
    <| condition c |>
    retryObl a (rightBranch c)
  | isObligation c =
    if action c==a then (success << c >> c) else c
  | ...

```

Contract interpretations: Access to the structure of contracts allows us to generate different interpretations, including monitoring of a trace according to a contract, output to external analysis tools and natural language generation. As to natural language generation, as yet we only have an *ad hoc* translator, but we eventually plan to reuse the grammars used for generation.

Syntactic sugar: In order to retain syntactic features of the controlled natural language operators which are expressed using syntactic sugar, whenever possible, a *deep embedding* is used — with higher level reasoning done using *algebraic manipulation* using the abstract operators (e.g. look for conflicts, and if any are discovered, explain them in terms of natural language) while using lower level reasoning at the basic logic level obtained by opening up the syntactic sugar (e.g. just reporting whether a contract is satisfiable or not). Essentially the approach is to keep the abstract operators for as long as possible and take the one-way road down to the basic logic only when we cannot do otherwise.

5 Remarks on a Possible Controlled Language

5.1 Some Key Examples

In this section we take the examples of section 1 and propose some simplifications which reduce both syntactic complexity and, more importantly, the potential for ambiguity.²

1. **original:** Upon accepting a job, the system guarantees that the results will be available within an hour unless cancelled in the meantime.

² At this exploratory stage such simplifications are guided primarily by our own intuitions rather than through conformity to any existing CNL that includes deontic operators (e.g. SBVR structured English).

controlled: if SYSTEM accepts Job, then during one hour it is obligatory that SYSTEM make available results of Job unless SOMEONE cancels Job.

logic: $\Box(\text{accept}_j \rightarrow O(\text{system} : (\text{result}_j + \text{cancel}_j))_{[0,1hr]})$

comment: There are three events: accepting a job, results being available, and a cancellation. There is also a contractual obligation concerning the second event, but this is discharged if the cancellation takes place. This sentence displays the classical problem of *attachment ambiguity*. Typically this concerns the point of application of a modifier expressed by a prepositional phrase. The classical example is “I shot an elephant in my trousers” where the modifier “in my trousers” can either apply to elephant (unlikely, given the physical dimensions involved) or to the shooting event.

In the case at hand the analogous problem is the attachment of the time adverbial *within an hour* which could be to the availability of the results, or to the obligation concerning the availability of results (cf. *within an hour I promise to go* vs. *I promise to go within an hour*).

There is also a problem of ellipsis: in the phrase “unless cancelled in the meantime”, the syntactic object of the the cancellation has been omitted, and this creates further ambiguity, since it could be the job or the results. We should note that in section 3, both these ambiguities have been resolved: attachment is to the obligation has been favoured, whilst the object of the cancellation is assumed to be the job.

Several problems can be solved by allowing proper names into the language. Predefined ones, like SYSTEM, are notated using all capital letters, whilst arbitrary ones, like Job, are used to enforce coreference, have an initial capital letter. A second major change is a rationalised event syntax based on a simple agent-action-object format. The easy cases are underlined above. The complex case revolves around *is obliged*, which can usefully be treated as an event (strictly it is a state) whose object is itself an event. We have carefully controlled the syntax and placement of the time adverbials, and, last but not least, the position of *unless* is immediately after the statement of the obligation.

The phrase up to the first comma defines an interval according to a strictly controlled syntax for time expressions. During that interval there are the two possible outcomes. The event syntax has also been rationalised to a simple agent-action-object format.

2. **original:** Only the owner of a job has permission to cancel the job.

controlled: it is permitted that only owner of Job cancels Job.³

logic: $\Box(P(\text{owner}_j : \text{cancel}_j) \ \& \ F(\overline{\text{owner}}_j : \text{cancel}_j))$

comment: A tricky issue in this sentence is the anaphoric status of the determined noun phrases “a job” and “the job”. These somehow have to be tied to referents. Informally, “a job” does not have a referent, so a referent is created, and this same referent is referenced by “the job”. Although it would be possible, following Fuchs-et-al. [13], to deal with this using Discourse Representation

³ The CNL grammar avoids the complexity of natural language quantifiers by treating “only” as a simple determiner.

Theory ((cf. Kamp and Reyle [14]), the introduction of proper names, notated in the CNL with an initial capital, offers a simpler solution since it allows the user to establish the coreference explicitly.

We should also note that the syntactic subject of “cancel” is missing. However, it is understood to be the same as the subject of “has permission” as a result of the syntactic behaviour of the verb “to have permission”. For example, if “John has permission to leave”, then John is the subject of leave. This behaviour is known as *subject-control* and can be handled within the framework of unification grammar. However, our CNL version of this sentence makes use of the “it is permitted that. . .” formulation in which the object of the permission is fully specified as an agent-action-object triple.

Another aspect concerns the initial noun-phrase, the syntactic structure of which includes “only” as a specifier. Note that semantically this usage makes sense in relation to a set of *referents* (here a singleton: the owner) and a set of *alternative referents* (here a complement set of other agents who might otherwise cancel the job), as reflected in the logical representation suggested in section 3

$$\Box(P(\text{owner}_j : \text{cancel}_j) \ \& \ F(\overline{\text{owner}}_j : \text{cancel}_j))$$

which is straightforward to engineer using techniques from unification grammar.

3. **original:** The system is forbidden from producing a result if it has been cancelled by the owner.

controlled: If owner of Job cancels Job, it is forbidden that SYSTEM produces result of Job

logic: $\Box(\text{cancel}_j \rightarrow F(\text{system} : (\Diamond(\text{result}_j))))$

comment: The main problem with the original sentence is that it the syntax is complex, and the word *it* ambiguously refers to either *the system* or *result* or *producing a result*, or, if we consider all three sentences to be part of a discourse, the job mentioned in the second sentence. Although it has been argued that this particular case could be resolved by semantic constraints or or by the use of heuristics (cf. ACE [13]), we feel that all these problems can be avoided by adopting the controlled language being suggested.

5.2 The Grammar

We are currently designing a grammar using PC-PATR, a version of the PATR2 (Shieber [15]) formalism that is available free of charge from the Summer Institute of Linguistics ⁴.

Such unification grammar formalisms facilitate the transformation of parse tree fragments shown into attribute-value structures that can be regarded as notational variants of the formulae presented in section 3 of this paper. The grammar is currently in the design phase, and our starting point has been the sentences discussed in section 5.1.

⁴ See <http://www.sil.org>

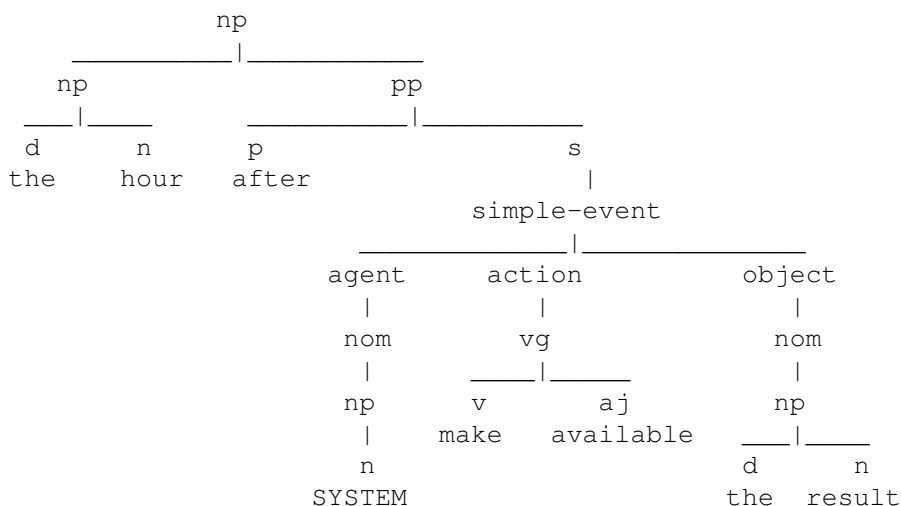
A central idea is that a sentence can describe a *simple event* and that an unadorned version of such an event — syntactically and semantically — is a agent-action-object triple of the kind used by the Semantic Web community as a building block within RDF⁵.

```
s          -> simple-event
simple-event -> agent action object
```

Typically an agent is nominal entity of some kind, an action is described by a verb-group, and an object is another nominal entity, although we allow for a sentential object in order to handle the object of verbs like “allow”, “prohibit” etc.

```
agent  -> nom
action -> vg
object -> nom | s
```

The simplest kind of nominal is a *noun phrase* which can reduce to a proper noun. More complex kinds of noun phrase can involve determiners, prepositional phrases, and hence, subordinate sentences. For example “the hour after SYSTEM make available the result”



Certain adornments to simple events, such as time adverbials are foreseen. More general adverbials are possible at the end of the simple event⁶.

```
s -> t-adv simple-event
s -> simple-event adv
```

⁵ RDF is a language for expressing simple propositions - see <http://www.w3.org/RDF/>

⁶ An adv reduces to a t-adv

Time adverbials can also range from simple one-word specification (“tomorrow”) to complex constructions involving subordinate sentences such as “after SYSTEM make available the result”.

In order to handle modalities we also include a rules of the following type

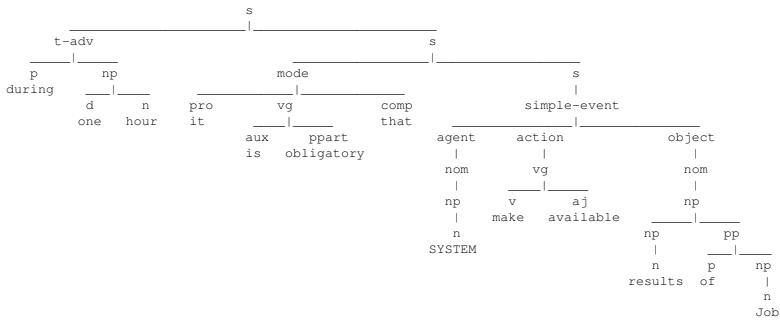
s -> mode s_1

where `mode` is defined to include phrases such as “it is forbidden/obligatory that” etc.

To conclude this section, we illustrate these ideas in context with two examples of parse trees for key example sentences 1 and 3:

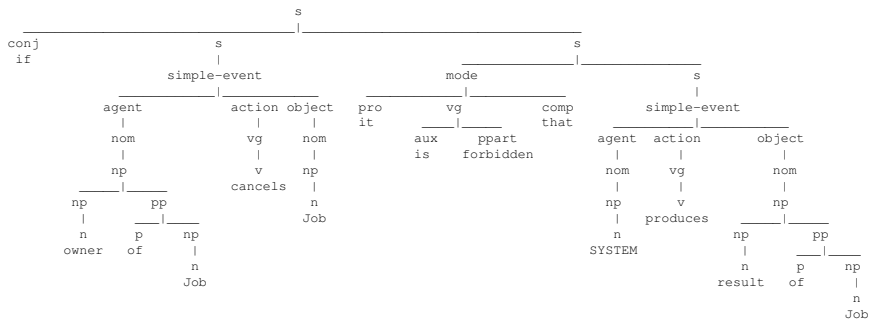
PART OF EXAMPLE 1

=====



EXAMPLE 3

--	--	--	--	--	--	--	--	--

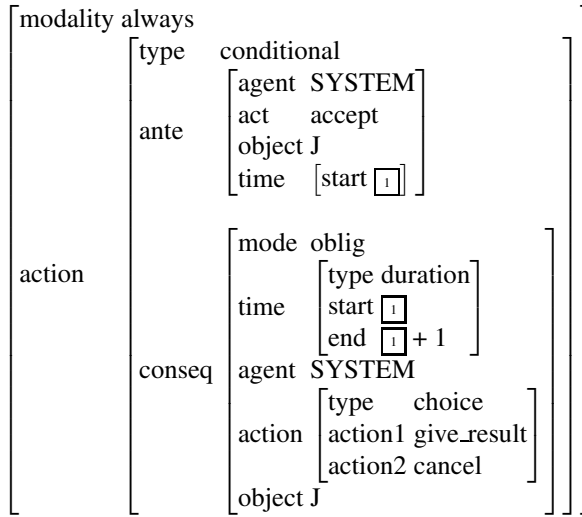


6 Semantic Constuction

The last section outlined the grammar which builds syntactic trees out of sentences. The next phase is to build logic expressions. The idea is to use constraint equations to build representation of such expressions expressed as feature structures which are essentially attribute-value matrices. Example 1, for instance has the following logical representation:

$$\Box(\text{accept}_j \rightarrow O(\text{system} : (\text{result}_j + \text{cancel}_j))_{[0,1hr]})$$

but this can also be expressed as a feature structure of the following kind:



Such structures are built by annotating grammar rules with constraint equations which the system attempts to satisfy as the grammar rules are invoked.

A successful parse then yields a feature structure of the above type as a possible solution to the constraint equations that have been satisfied. To give a very intuitive example of this, the first line of the above representation could be produced by the following annotation on the respective grammar rule:

```
s -> mode s_1
<s modality> = <mode type>
```

In other words, the value of the modality attribute of the *s* is the same as the type attribute of the mode – which in this case would be the constant “always”.

7 Conclusions and Future Work

In this article we have outlined the proposal for a controlled natural language and a deontic logic to enable specification and reasoning about contracts in an automatic fashion. A number of challenges still remain to be addressed from both linguistic and logic perspectives.

On the linguistic side the priorities are to firm up the existing grammar specification and to write the rule annotations which will carry out the semantic construction. If this is done correctly, the “translation” between the feature structure and logic representation will be trivial because they should be notational variants.

On the logic side, we plan to use contract analysis techniques similar to ones we have recently developed for CL to enable contract sanity checking, including discovery of conflict analysis and superfluous clauses. These techniques can provide feedback to

the contract translation from the natural sublanguage into logic, for example, for the resolution of ambiguities.

The main aim of transforming logic to language is to make it more understandable to those unfamiliar with logic. The transformation is clearly a problem of Natural Language Generation (NLG) which is generally recognised (cf. Reiter and Dale [16]) to involve large numbers of realisation choices. These can of course be by-passed by adopting a suitably strict generation algorithm. The extent to which this compromises naturalness is a factor that will need to be carefully evaluated in the contracts domain.

Once a basic system is in place we will start to experiment with CNL generation tasks, the very first of which will be to translate a logic representation of a contract back into CNL. This will then be compared to the original contract. In parallel we will define some simple contract-related inference tasks which will result in new logic expressions. A challenge will then be to investigate the issues involved in adapting the generation algorithm so that the system will report the inferences made back to the user using CNL.

References

1. Pulman, S.: Controlled language for knowledge representation. In: *Proceedings of the First International Workshop on Controlled Language Applications*, Leuven, Belgium (1996)
2. Vauquois, B.: A survey of formal grammars and algorithms for recognition and transformation in mechanical translation. In: *IFIP Congress*, vol. (2), pp. 1114–1122 (1968)
3. Meyer, J.J.C., Dignum, F., Wieringa, R.: The paradoxes of deontic logic revisited: A computer science perspective (or: Should computer scientists be bothered by the concerns of philosophers?). Technical Report UU-CS-1994-38, Department of Information and Computing Sciences, Utrecht University (1994)
4. Pace, G.J., Schneider, G.: Challenges in the specification of full contracts. In: Leuschel, M., Wehrheim, H. (eds.) *IFM 2009*. LNCS, vol. 5423, Springer, Heidelberg (2009)
5. Prisacariu, C., Schneider, G.: A formal language for electronic contracts. In: Bonsangue, M.M., Johnsen, E.B. (eds.) *FMOODS 2007*. LNCS, vol. 4468, pp. 174–189. Springer, Heidelberg (2007)
6. Pace, G., Prisacariu, C., Schneider, G.: Model checking contracts – a case study. In: Namjoshi, K.S., Yoneda, T., Higashino, T., Okamura, Y. (eds.) *ATVA 2007*. LNCS, vol. 4762, Springer, Heidelberg (2007)
7. Asarin, E., Caspi, P., Maler, O.: Timed regular expressions. *Journal of the ACM* 49(2), 172–206 (2002)
8. Hudak, P.: Building domain-specific embedded languages. *ACM Computing Surveys* 28, 196 (1996)
9. Hudak, P.: Modular domain specific languages and tools. In: Devanbu, P., Poulin, J. (eds.) *Proceedings of the 5th International Conference on Software Reuse*, pp. 134–142. IEEE Computer Society Press, Los Alamitos (1998)
10. Jones, S.P., Eber, J.M., Seward, J.: Composing contracts: an adventure in financial engineering (functional pearl). In: *ICFP 2000: Proceedings of the 5th ACM SIGPLAN international conference on Functional programming*, pp. 280–292. ACM Press, New York (2000)
11. Claessen, K., Sheeran, M., Singh, S.: Functional hardware description in Lava. In: *The Fun of Programming. Cornerstones of Computing*, Palgrave, pp. 151–176 (2003)
12. Jones, S.P.: *Haskell 98 Language and Libraries: The Revised Report*. Cambridge University Press, Cambridge (2003)

13. Fuchs, N.E., Kaljurand, K., Kuhn, T.: Attempto Controlled English for Knowledge Representation. In: Baroglio, C., Bonatti, P.A., Małuszyński, J., Marchiori, M., Polleres, A., Schaffert, S. (eds.) Reasoning Web. LNCS, vol. 5224, pp. 104–124. Springer, Heidelberg (2008)
14. Kamp, H., Reyle, U.: From Discourse to Logic: Introduction to Model-theoretic Semantics of Natural Language, Formal Logic and Discourse Representation Theory. Studies in Linguistics and Philosophy. Springer, Heidelberg (1993)
15. Shieber, S.: An Introduction to Unification-Based Approaches to Grammar. CSLI Publications, Stanford University (1986)
16. Reiter, E., Dale, R.: Building natural language generation systems. Studies in natural language processing. Cambridge University Press, Cambridge (2000)

Rabbit to OWL: Ontology Authoring with a CNL-Based Tool

Ronald Denaux¹, Vania Dimitrova¹, Anthony G. Cohn¹, Catherine Dolbear³,
and Glen Hart²

¹ School of Computing, University of Leeds, Woodhouse Lane, Leeds, LS2 9JT, UK
`r.denaux@leeds.ac.uk`

² Ordnance Survey Research, Romsey Rd, Southampton, SO16 4GU, UK

³ Sharp Laboratories of Europe Limited, Edmund Halley Rd, Oxford Science Park,
OX4 4GB, UK

Abstract. Recent work on ontology engineering has seen the adoption of controlled natural languages to ease the process of ontology authoring. However, CNL-based tools still require good knowledge engineering skills to be used efficiently. In this paper presents ROO, an ontology authoring tool that has been designed to cater for the needs of domain experts with little or no ontology engineering experience. ROO combines a CNL-based interface with appropriate tool support based on an ontology construction methodology. We focus on how this tool support is provided in ROO by using and implementing novel aspects of the Rabbit controlled natural language and we refer to an evaluation study that provides empirical evidence in support of using CNL-based techniques to assist ontology authors.

1 Introduction

There is a recent trend of using controlled natural language (CNL) interfaces to provide more intuitive ways for entering abstract knowledge constructs [1]. This can reduce the complexity of knowledge formulation, which can lead to wider user involvement in ontology authoring and improved efficiency of the knowledge engineering process. However, CNL-based tools for ontology engineering focus solely on providing a CNL interface, while ignoring the *whole* ontology construction process, and still require good knowledge engineering skills. We have developed a novel approach where a CNL-based tool has been designed to support the involvement of domain experts without knowledge engineering background in the overall ontology authoring process. This work has been inspired by the ontology authoring experience at Ordnance Survey, the mapping agency of Great Britain.

The Ordnance Survey is developing a modular topographic domain ontology to facilitate the description and reuse of its topographic data by third parties [2]. At the heart of ontology development is the *active involvement of domain experts* (e.g. geographers and ecologists), which is reflected in the Ordnance Survey's methodology for ontology construction [2], comprising of several steps:

- Identifying the scope, purpose and other requirements of the ontology;
- Gathering source knowledge and documents and identifying ontologies for reuse;
- Capturing the ontology content in a knowledge glossary;
- Formally defining core concepts and relationships between concepts by using structured English sentences;
- Converting the structured English sentences into OWL¹;
- Ontology verification and validation.

Following this methodology, the domain expert is engaged in the construction of a *conceptual ontology* which involves the first four steps. The knowledge engineer then performs the last two steps, which focus on the logical level of the ontology.

A crucial component of the Ordnance Survey methodology is the use of a controlled language for authoring the conceptual ontology — a CNL, called *Rabbit*, has been developed for this purpose [3]. The aim is for *Rabbit* to be easy to read and write by domain experts, allowing them to express what they need to in order to describe their domain. The design and evaluation of *Rabbit* is presented elsewhere [4], while a comparison with other controlled languages is given in [5].

Our main goal is to investigate whether it is possible for domain experts to build ontologies without having to take a course on ontology engineering and tools such as Protégé². Instead, we aim to provide a tool that can be used with minimal training and results in OWL ontologies that represent the domain expert's understanding of the domain. This paper describes our approach, which combines (i) *Rabbit*, a CNL that has been designed to be authored by domain experts (Sect. 3) and (ii) ROO³, a tool that provides support for ontology construction and editing of CNL sentences (Sect. 4). This paper focuses on the support for creating OWL ontologies in *Rabbit*, so we illustrate how specific support for domain experts is achieved (this requires a more detailed description of the *Rabbit* parser implementation, see Sect. 5). Finally, we describe an evaluation of our approach (Sect. 6) and provide a summary and conclusions in Sect. 7.

2 Related Work

In the context of the Semantic Web, *Rabbit* [4], CLOnE [6], ACE [7], and SOS [8] are all CNLs that can be used to view, create and edit OWL ontologies. We are not aware of any tool support for the Sydney OWL Syntax, although there is some tool support for PENG [9] that forms the basis for SOS. CLOnE only

¹ OWL (Web Ontology Language) is a W3C standard for authoring ontologies intended to be used in the Semantic Web. See <http://www.w3.org/TR/owl-features/>

² <http://protege.stanford.edu/>

³ ROO is developed within the Confluence project and is distributed as open source. It can be downloaded from: <http://www.comp.leeds.ac.uk/confluence/>

supports a small number of the axioms of OWL, so only lightweight ontologies can be built using CLOnE.

ACE is the most mature CNL having originally been created to translate into First Order Logic. A subset of ACE is now used to drive applications such as ACE View [10] where the resulting sentences are translated into OWL and SWRL. ACE View and ROO, the solution described in this paper, were developed at the same time. ACE View and ROO both provide a CNL interface to enter and view ontology knowledge that is automatically translated into OWL. The main difference between ACE View and ROO is that ACE View still requires knowledge engineering expertise in order to be used effectively. This is due to the lack of guidance through the ontology construction process, which is missing in ACE View and is provided by ROO because it focuses on domain experts and novice users who have no previous experience with ontology construction. Another important difference between ACE View and ROO is the number of ontology construction tasks that are supported. At the time of writing, ROO supports the authoring and inspection of ontologies using a CNL interface. ACE View provides a wider range of task support by also allowing translation from OWL into CNL, which enables the inspection of inferences. We performed a comparative evaluation study between ACE View and ROO, which showed that a CNL interface alone is not enough to properly support domain experts who lack knowledge engineering experience (see section 6).

Another mature CNL is CPL, developed at Boeing, and used by the HALO project [11,12]. HALO improved over other CNL approaches by providing a more holistic approach: the CNL is provided in conjunction with support of the ontology construction process and not just as a standalone tool for entering knowledge into the system. Due to the large scope of that project, HALO focused on query answering and using a wide variety of techniques such as information extraction to build the ontology. It is therefore not explicit how the CNL aspect combined with tool support aids domain experts to abstract the domain and reformulate their knowledge from natural language into the CNL, which is our main focus.

CNL interfaces have also been proposed for query answering [11,13,14,15]. These languages make it easier for domain experts to evaluate an ontology, as they can pose questions that need to be answered based on the ontology and a set of instances. These languages cannot be used to construct ontologies because they do not support the input of new knowledge into the ontology.

3 The Rabbit Controlled Natural Language

A central part of our proposed solution to enable domain experts to understand and author ontologies, is the use of a controlled natural language. The Ordnance Survey has designed Rabbit [16] to be easy for domain experts to digest and produce, allowing them to express what they need to in order to describe their domain⁴.

⁴ The Ordnance Survey named the Rabbit language after Rabbit in Winnie the Pooh, who was actually cleverer than Owl.

The fundamental principles underlying the design of *Rabbit* are:

- To allow the domain expert to express their knowledge as easily and simply as possible and in as much detail as necessary. It is written to appear like a natural English statement;
- To have a well defined grammar and be sufficiently formal to enable those aspects that can be expressed as OWL to be systematically translatable;
- To recognize that the domain expert alone cannot produce an ontology and that a knowledge engineer is also necessary;
- To be used in conjunction with tools which help to enforce an authoring method but not to the point where *Rabbit* is only readable through tools;
- To be domain independent.

Tables 1, 2 and 3 show commonly used sentence structures for describing concepts, relationships and individuals. The tables also show the translation of the *Rabbit* sentence in the OWL Manchester Syntax [17]. Note that the tables show most of the *Rabbit* constructs, but cannot show all possible combinations of these constructs for reasons of space. The language covers most of the constructs in OWL 2, see [16]. Ordnance Survey has performed user tests to investigate *Rabbit* sentence comprehension and ease of authoring without tool support [4,18].

This paper focuses on how we used the *Rabbit* language specification to provide tool support for domain experts. In a different research strand, the language itself was designed and evaluated (see [4] for detail). In the following sections, we describe the implementation of a parser based on the *Rabbit* specification and the role of that parser in providing a user interface that is easy to use for domain experts.

4 ROO: Rabbit to OWL Ontology Authoring Tool

ROO is an ontology creation tool based on Protégé 4 that assists domain experts to build conceptual ontologies. Domain experts are people who (i) have extensive knowledge of the domain to be modelled; (ii) are used to thinking at an abstract level about the domain (so they can provide descriptions and knowledge sources for concepts and relationships of the domain); (iii) have no previous experience building ontologies, but may have experience building other types of models in their specific domain; (iv) have no interest in becoming knowledge engineering experts or experts in logical formalisms; (v) have a fairly well defined idea of how the resulting ontology will be used.

ROO uses *Rabbit* to enable domain experts to automatically formalise their knowledge in OWL. ROO provides easy to understand suggestions and task-specific messages to help the user enter correct CNL constructs. Appropriate feedback is given to help users recognise concepts, relationships and individuals when writing CNL sentences. Syntax highlighting based on the parsed structure helps the user recognize CNL patterns.

Table 1. Example Rabbit sentences for describing concepts. Source: the Ordnance Survey hydrology ontology [19].

Description	Rabbit	Manchester Syntax
Concept Declaration	Stream is a concept.	Class: Stream SubClassOf: owl:Thing
Subjunction	Every Cataract is a kind of Waterfall.	Class: Cataract SubClassOf: Waterfall
Defined Class	A Source is anything that: is a kind of Spring or Wetland; feeds a River or a Stream.	Class Source: EquivalentTo: Spring or Wetland and feeds some (River or Stream)
Existential Quantifier	Every River flows into a Sea.	Class: River SubClassOf: flowsInto Some Sea
Negation	No Backwater has a Current.	Class Backwater: DisjointWith: have some Current
Union	Every Bifurcation is part of one or more of River or Stream.	Class: Bifurcation SubClassOf: bePartOf some (River or Stream)
Minimal Cardinality Restriction	Every Confluence flows from at least two River Stretches or Streams.	Class: Confluence SubClassOf: (flowFrom min 2 (RiverStretch or Stream))
Qualified Cardinality Restriction	Every Channel has exactly 2 Banks.	Class Channel: SubClassOf: hasBank exactly 2 Bank
Universal and Existential Quantifiers	Every River only flows into a Sea.	Class River: SubClassOf: flowsInto some Sea and flowsInto only Sea
Built-in Properties for Anonymous Classes	Every Reservoir has purpose Storage of Water.	Class: Reservoir SubClassOf: havePurpose some (Storage and (Rabbit:of some Water))
Complex Objects	Every Spout gushes Water that flows from the Ground.	Class: Spout SubClassOf: gush some (Water) and (flowFrom some Ground))
Disjointness	Canal and Disused Canal are mutually exclusive.	Class: Canal DisjointWith: DisusedCanal
Modality	A Broad usually is located in East Anglia.	no translation
Homonym	Dam (Water) is a concept.	Class: Dam_Water
Disambiguation	Dam (Structure) is a concept.	Class: Dam_Structure

Table 2. Example Rabbit sentences for describing instances. Source: the Ordnance Survey hydrology ontology [19].

Description Rabbit		Manchester Syntax
Instance Declaration	Somerset is a County.	Individual somerset: Types: County
Same Instance	UK and United Kingdom are the same thing.	Individual: uk SameAs: unitedKingdom
Different Instance	River Wharfe and River Aire are different.	Individual: riverWharfe DifferentFrom: riverAire

Table 3. Example Rabbit sentences for describing relationships. Source: the Ordnance Survey hydrology ontology [19].

Description Rabbit		Manchester Syntax
Relationship Declaration	enables is a relationship.	ObjectProperty: enable
SubProperty	The relationship flows in is a special type of the relationship is connected to.	ObjectProperty: flowIn SubPropertyOf: beConnectTo
Inverse Relationship	The relationship is fed by is the inverse of feeds.	ObjectProperty: beFeedBy InverseOf: feed
Range Restriction	The relationship is capital city of must have the object Country.	ObjectProperty: beCapitalCityOf Range: Country
Functional Property	The relationship is capital city of can only have one object.	ObjectProperty: beCapitalCityOf Characteristics: Functional

ROO avoids using technical terminology, preferring to use conceptual terminology that may not be well-defined in a technical sense, but which is easier to understand for novice users. In the case of OWL, ROO will avoid introducing terminology such as *Object Property*, opting to use *relationship* instead. In the case of the controlled natural language, ROO avoids using linguistic terminology such as *determiner* as much as possible. When technical terminology is introduced, ROO tries to give specific examples, preferably coming from the domain (i.e from the ontology itself).

ROO helps users to avoid introducing ambiguity in the resulting ontology. The CNL tool avoids making assumptions by requiring the declaration of concepts, relationships and individuals. ROO is aware of cases when parsed sentences could be ambiguous and warns the user accordingly, preferably suggesting ways to avoid ambiguity.

ROO provides guidance about how to build ontologies that are easy to reuse. ROO incorporates a model of Kanga [2], an Ontology Engineering methodology, and can make suggestions to the user regarding tasks that need to be performed to improve the reusability and general quality of the ontology.

ROO provides easy access to documentation to explain the user interface, the controlled natural language and the ontology creation process.

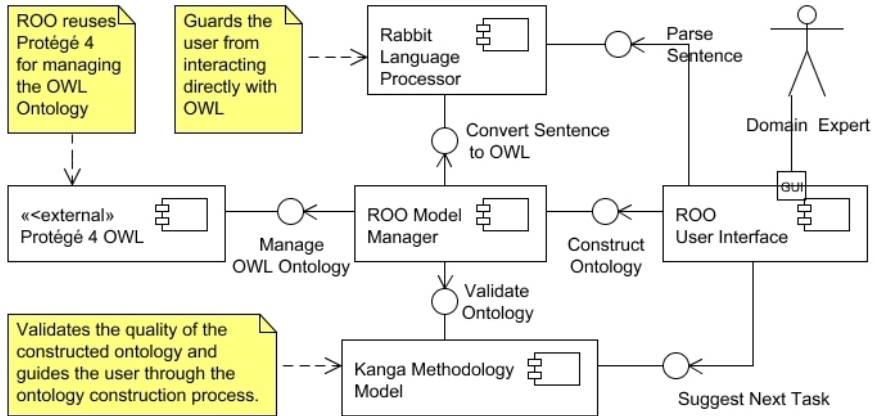


Fig. 1. UML 2.0 functional architectural view of ROO shows the architectural elements, interfaces and inter-element connections

4.1 Architectural Overview

Fig. 1 shows the main architectural elements of ROO, and how these elements interact with each other. Here, we introduce each architectural element.

The user of ROO interacts with the *ROO User Interface*, which is a collection of components that enable domain experts to create and edit ontologies by writing Rabbit sentences. The user interface uses services provided by the Rabbit Language Processor (to parse sentences), the ROO Model Manager (to construct the ontology) and the Kanga Methodology Model (to suggest ontology construction tasks to the user).

The *Rabbit Language Processor* consists of a Rabbit parser – that parses text into an unambiguous intermediate representation– and a Rabbit to OWL converter – that converts the unambiguous intermediate representation into OWL. The ROO Model Manager and the ROO User Interface use the services provided by the Rabbit Language Processor.

The *ROO Model Manager* acts as a central point between the underlying OWL ontology and the other architectural elements. It provides methods to construct the ontology by accepting Rabbit sentences and uses the Kanga Methodology Model to validate that the ontology complies with the rules set by Kanga Methodology. The ROO Model Manager is implemented as a thin layer on top of Protégé 4 OWL, which already provides services to manage OWL ontologies.

The *Kanga Methodology Model* provides services to check whether a given OWL ontology contains annotations and entities that would be expected if the ontology was built using the Kanga Methodology for ontology construction. If the ontology is missing annotations or entities, the Kanga Methodology Model can suggest ontology construction tasks from the Kanga Methodology. In ROO, this architectural element has been implemented as a rule based system in JBoss Drools⁵.

⁵ <http://jboss.org/drools/>

In this paper, we focus on the **Rabbit** Language Processor and the part of the ROO User Interface that is used to edit **Rabbit** sentences. More specifically, in the next sections we show how CNL languages can support domain experts with little knowledge engineering experience by highlighting design and implementation decisions in the **Rabbit** Language Processor and the ROO User Interface.

5 Providing Domain Expert-Specific Tool Support for **Rabbit**

This section gives an overview of how the **Rabbit** parser and OWL generator are implemented (section 5.1), gives a detailed description of how ambiguity is used by the **Rabbit** parser to provide extra support for domain experts (section 5.2) and explains how the parser implementation is used to provide a user interface that domain experts can learn quickly (section 5.3).

5.1 **RabbitParser**

The **Rabbit** parser consumes a document containing **Rabbit** sentences and produces a parse tree (an intermediate representation of the document). If the parsed document contains sentences which are not valid **Rabbit** sentences, the parser marks these sentences and attaches error messages that should help the users to correct the sentence.

The **Rabbit** parser consists of a pipeline of linguistic Processing Resources as pictured in Fig. 2. The implementation of the parser included in ROO follows the CLonE [6] approach and is based on the GATE⁶ text processing environment. Fig. 2 provides an overview of the pipe-line (the current pipeline consists of around 30 Processing Resources) and shows the three main phases of the parsing process:

First, there is a pre-processing phase where we use GATE — more specifically ANNIE[20, Chapter 8] — to perform the basic natural language pre-processing such as tokenization, sentence splitting, Part of Speech (POS) and morphological tagging.

The second major phase in parsing **Rabbit** includes using a gazetteer to find **Rabbit** key phrases and using JAPE⁷ transducers to find and process **Rabbit** constructs based on the annotations gathered during the pre-processing phase. Roughly speaking, there is one JAPE transducer per BNF rule in the **Rabbit** grammar as shown in table 4, although in some cases, our implementation deviates from the **Rabbit** BNF in order to achieve a better parsing efficiency or in order to control the amount of ambiguity the user can introduce (see details in Sect. 5.2).

⁶ <http://gate.ac.uk>

⁷ Java Annotation Pattern Engine [20, Chapter 7] provides a language for finding annotation patterns during the parsing process. It also provides hooks for invoking Java methods during the parsing of a text.

as shown in tables 1, 3 and 2. These tables only show the minimum of axioms generated to express the OWL equivalent, in practice the generator also includes annotations such as `rdf:labels` and `rabbit:sentence` which make the resulting ontology easier to read (using ROO or Protégé).

Although the current implementation of the OWL generator uses the OWL API, the mapping from the parsed tree to the corresponding statements in OWL has been isolated by the API. Hence, creating an OWL generator using an alternative API, such as Jena, should be a trivial task.

Mapping Rabbit Entities to OWL Entities. A crucial step during the parsing and validation of Rabbit sentences is to link Rabbit Entities (`IParsedConcept`, `IParsedRelation` and `IParsedInstance`) to OWL Entities (OWL Classes, OWL Object Properties and OWL Individuals). The current implementation follows the CLOnE [6] approach by implementing a canonicalisation procedure to create a key identifier for the entities. This canonicalisation uses the root morpheme of the words comprising a Rabbit entity to create an OWL id. The third sentence in Table 3 illustrates the results of this canonicalisation process as the Rabbit relationship `is fed by` has been mapped to the canonical name `beFeedBy`. Although `beFeedBy` is not easy to read, the OWL generator also adds the `rdfs:label "is fed by"` annotation (not shown in table 3), which is used by most applications instead of the OWL id.

A possible disadvantage of the canonicalisation process is that users could write ungrammatical sentences (e.g. `Every Confluences flows from at least two Rivers Stretch or Stream`). However, none of the practical experiences with ROO users had this problem.

The advantage of generating class identifiers based on the root morpheme is that the natural language processing engine (GATE, in this case) takes care of morphology variations, making it easier for domain experts to write sentences in the way they would when writing natural language. For example, users write `Every Confluence flows from at least two River Stretches or Streams`, but only need to declare that `River Stretch is a concept` and `Stream is a concept`, without needing to specify the plural forms. By using the morphological annotations added by ANNIE during the pre-processing stage of Rabbit parsing (see Sect. 5.1), the parser maps `River Stretches` and `River Stretch` to the same OWL class.

5.2 Handling Ambiguity in Rabbit Sentences

Most constructs in Rabbit contain key phrases which makes the constructs unambiguous. For example: `River Wharfe and River Aire are different`, where we have underlined the key phrases. The parser can search for text that matches the pattern `<individual> and <individual> are different`⁹.

In order to permit users to give natural names to concepts, relationships and instances, Rabbit allows these constructs to consist of multiple words. Examples of concepts that should be valid are: `Natural Body of Water`, `Water`

⁹ An `<individual>` can be any number of tokens (excluding keyphrases).

for Irrigation, Fire and Rescue Services and Formula 1. At the same time, the following relationships are also valid: **has pet**, **flows into**, **contains**, **contains water for**, **Services** (as a relationship meaning “to provide services to”) and **Rescues**. This freedom to name Rabbit entities allows users to construct more natural sentences and to use concepts and relations at an appropriate granularity level for their domain. At the same time, this freedom has the potential to introduce ambiguities. The Rabbit parser and ROO provide support to avoid introducing ambiguity.

Table 5. Handling ambiguity in Rabbit. The table shows how the sentence **Every Irrigation Canal contains Water for Irrigation.** is interpreted depending on the context of Rabbit sentences.

ID	Context Sentences	Manchester Syntax Translation or Error Message when translating sentence Every Irrigation Canal contains Water for Irrigation.
1	no context	2 errors: missing concepts Irrigation Canal and Water for Irrigation 1 error: missing relationship contains 1 error: alternative interpretation for Water for Irrigation' 1 error: alternative interpretation for 'contains' 'Water for Irrigation'
2	Irrigation Canal is a concept. contains is a relationship. Water for Irrigation is a concept.	Class: IrrigationCanal SubClassOf: contain some WaterForIrrigation
3	Irrigation Canal is a concept. contains is a relationship. Water is a concept. Irrigation is a concept.	Class: IrrigationCanal SubClassOf: contain some (Water and (Rabbit:for some Irrigation))
4	Irrigation Canal is a concept. contains water for is a relationship. Irrigation is a concept.	Class: IrrigationCanal SubClassOf: containWaterFor some Irrigation
5	Irrigation Canal is a concept. contains is a relationship. Water is a concept. Irrigation is a concept. Water for Irrigation is a concept.	1 error: 'Water for Irrigation' has an equally possible alternative interpretation 'Water' for 'Irrigation'

Table 5 shows an example of how a Rabbit sentence can be interpreted in different ways depending on its context. The context of a Rabbit sentence is defined by the set of Rabbit sentences that have been entered for the ontology. Context 1 in table 5 is the empty context. In this case the parser chooses an interpretation where it expects **Water for Irrigation** to be a concept and **contains** a relationship. However, the parser also has recognised that **Water for Irrigation** could interpreted differently as being composed of concepts **Water** and **Irrigation** and linked by the keyphrase **for**. This shows that the parser keeps alternative interpretations in the parse tree and uses these to warn and prevent the user from introducing ambiguity.

The different contexts in table 5 illustrate how the parser chooses the correct interpretation based on the context. For example, in order to decide whether **Water for Irrigation** is a single concept or a Rabbit *compound object* (two concepts linked by the built-in relationship **for**) the parse tree validation uses

a disambiguation algorithm that ranks each parsing option based on the entities declared in the context. So, if the user has declared concepts **Water** and **Irrigation**, but not **Water for Irrigation** (as in context 3 in table 5), then the disambiguation algorithm will conclude that **Water for Irrigation** is more likely to be a compound object and not a single concept. Alternatively, if the user has declared concept **Water for Irrigation**, but not the concepts **Water** or **Irrigation**, then the disambiguation algorithm will rank the interpretation of **Water for Irrigation** as a single concept higher than as a compound object (see context 2 in table 5).

The disambiguation algorithm illustrated above is able to identify potentially ambiguous constructs and benefits the domain experts by giving them more freedom to use their own terminology when describing their domain. However, if the number of alternative interpretations is very large, the algorithm will find too many potentially ambiguous constructs which can have an adverse result: the users will not be able to introduce new terminology due to potential ambiguities. In order to restrict the number of potential ambiguous **Rabbit** constructs, we (i) require users to explicitly introduce entities by using the three **Rabbit** entity declaration sentences; (ii) only allow specific constructs in the **Rabbit** language to be ambiguous and (iii) impose linguistic restrictions on what are valid **Rabbit** concepts and relationships. We explain these three restrictions next.

Rabbit Entity Declaration Sentences. The **Rabbit** parser is able to recognise potential entities (which we call concept candidates, relationship candidates and instance candidates) when parsing sentences. For example, given sentence **Every River flows into a Sea.**, the parser will recognise that **River** and **Sea** are concept candidates and **flows into** is a relationship candidate. However, this sentence on its own is not a valid **Rabbit** sentence because the user has not explicitly introduced the required concepts and relationships. The parser will currently show error messages stating that **concept 'River' has not been introduced in the ontology yet**. The **Rabbit** language provides entity declaration sentences to introduce concepts (**<concept candidate> is a concept.**), relationships (**<relationship candidate> is a relationship.**) and instances (**<instance candidate> is a <concept>.**). The disadvantage of using entity declaration sentences is that users need to write more sentences at the beginning of the ontology construction process. The advantage is that the parser can then use the set of introduced entities to disambiguate sentence constructs. Furthermore, having an explicit declaration sentence forces users to consider whether an entity that is introduced is essential for describing the domain or not.

Ambiguous Rabbit Constructs. The **Rabbit** parser only allows two constructs to be ambiguous: **Objects** and **Relationship Phrases**. **Rabbit Objects** are translated as (anonymous) classes in OWL. Example **Objects** are: **River**, **Body of Water**, **Irrigation of Water** and **Building that has purpose Education**. **Objects** can be ambiguous because concept names allow prepositions (see Table 6 in Section 5.2), but **Rabbit** also provides built-in prepositions to combine two

concepts into an **Object**. For example: **Irrigation of Water**, can be an **Object** containing a single concept (introduced by sentence **Irrigation of Water is a concept.**), or it can be an **Object** that relates concepts **Irrigation** and **Water** using the built-in **Rabbit** relationship of.

The second potentially ambiguous construct in **Rabbit** is the **Relationship Phrase**, which translates as an anonymous class in OWL by combining a relationship and an **Object**. Example **Relationship Phrases** are: **has purpose Education** and **contains water for Irrigation**. Ambiguity is possible in this case due to the freedom **Rabbit** gives users to define concepts and relationships. For example, the user has the freedom to define relationship **has purpose**, or the more generic relationship **has**, which affects how the **has purpose Education** is interpreted.

Linguistic Restrictions on Rabbit Entities. The main intuition in linguistically restricting **Rabbit** concepts and relationships is that concepts tend to be nouns (**River**) or noun phrases (**Body of Water**); and relationships tend to be verbs (**contains**) or verb phrases (**flows into**). The parser implementation has JAPE rules that find noun and verb phrases which are then interpreted as possible concept or relationships in the **Rabbit** sentence. Table 6 shows the current set of rules with examples of valid concept and relationships.

Table 6. Rules for detecting noun and verb phrases in the **Rabbit** parser implementation. The rules use high-level linguistic definitions, which are themselves defined as rules that use the Part of Speech tag. So **Adverb** is any token that has POS RB or VBN (past participle is interpreted as an adverb to handle concepts such as **Written Article**). See the GATE User Manual [20, Appendix E] for the codes used by the POS tagger.

JAPE rule	Example
Rule: NounPhrase((Adverb)* (Adjective)* (((Noun) (SpecialToken))* (Noun) (SpecialToken)*)) : RabbitNounPhrase	Extremely Large River Peer 2 Peer Network Boeing 737-300 Fire & Rescue Services Written Article
Rule: CompoundNounPhrase ((Noun) (Prep) (Noun)) : RabbitNounPhrase	Body Of Water School Of Computing North By Northwest
Rule: VerbPhrase ((Verb)+ (Prep)? ({RabbitNounPhrase})? (Prep)?) : RabbitVerbPhrase	has has purpose produces sound by runs into sea at is close to

Rabbit Syntactic Disambiguation Mechanisms. Besides the three mechanisms to restrict the potential for ambiguity in **Rabbit** sentences, the language also provides two built-in syntactic mechanisms to avoid ambiguity. First, concept names can contain a disambiguation component. For example, the Ordnance Survey Hydrology Ontology defines two different concepts that have the same name **Dam**. A **Dam** refer to both the physical construction that holds water, or it

can refer to the water itself. In order to separate the two concepts, while still using the **Dam** name, Rabbit allows users to introduce the concepts as **Dam (Water)** and **Dam (Structure)**. The user has to use these names for all the sentences describing the concepts (e.g. **A Dam (Water) is a Body of Water.**).

The second syntactic disambiguation mechanism relies on the capability of importing and reusing external ontologies. Rabbit allows users to import an ontology by giving that ontology a **label**. For example **Use ontology: [Hydrology]** from <http://example.com/Hydrology.rbt>, where [Hydrology] is the label. In that case, the user can refer to concepts defined in the imported ontology as **Bank [Hydrology]**, which can be used to disambiguate a concept with the same name but from a different semantic definition (e.g. **Bank [Finance]**).

5.3 User Interface for Editing Rabbit

All the features described in the previous sections to make it easier for domain experts to build ontologies are made available through a customised user interface. The user interface is built on top of Protégé 4, but provides different tabs and components that use the Rabbit language instead of Manchester Syntax and class hierarchies. Fig. 3 shows the user interface of ROO. In this section we present some of the features that ROO provides to make it easier for domain

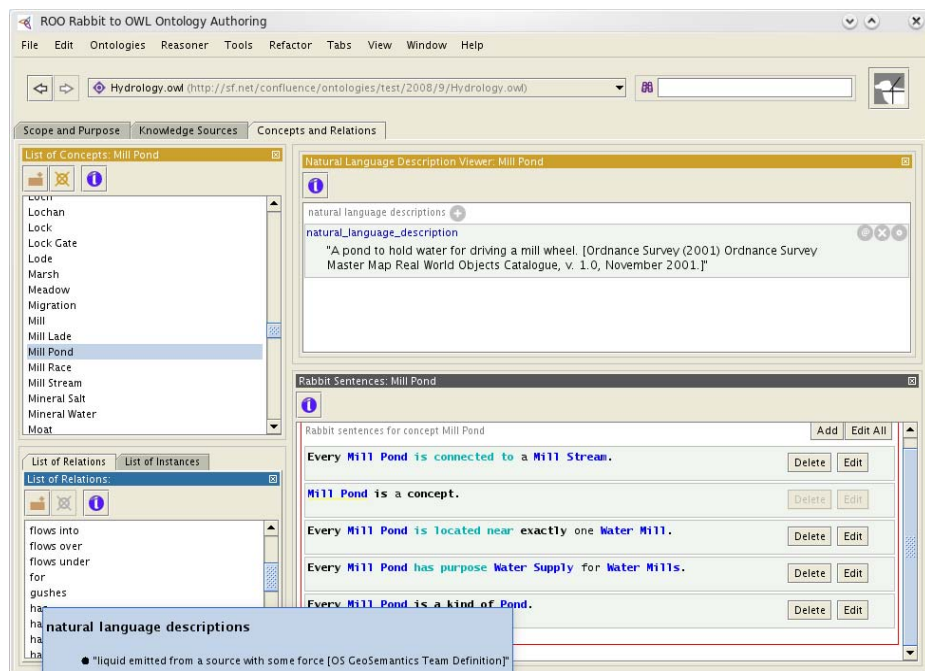


Fig. 3. Graphical user interface of ROO. ROO provides custom tabs and components to build ontologies that use terminology that is easy to learn by domain experts.

experts to create ontologies, focusing on those features that are related to the editing of Rabbit sentences.

The first thing users require for using ROO is to have easy access to documentation of the tool. Every component in the user interface provides a button that opens a browser and shows specific help about that user interface component and how it can be used to create ontologies. The documentation also includes an introduction to the Kanga methodology and the Rabbit language. The Rabbit language is presented in different ways to users who are at different stages of the Rabbit learning curve. As an introduction to the language we explain all the keywords in Rabbit, which includes descriptions of how to use keywords in sentences and example sentences. The documentation also provides a reference documentation for the Rabbit language, which presents sentences that can be used to accomplish ontology construction tasks such as *introduce a concept*, or *describe a relationship specialisation*. We also provide an introduction to Rabbit for people who already know Manchester Syntax and a set of example ontologies to help users get an idea of how to use Rabbit and ROO.

Once users have received an introduction to Rabbit, they can add knowledge to the ontology by writing Rabbit sentences. The user interface provides a Rabbit editor which provides syntax highlighting that uses the intermediate representation provided by the Rabbit parser. The syntax highlighter uses different colours that help the user see which parts of the sentence are recognised as concepts, relationships, instances, labels of imported ontologies and Rabbit keywords.

The Rabbit editor also gives feedback about syntax errors and warnings. This is done in three different ways: (i) as a list under the editor; (ii) by showing squiggly lines under the word (or words) causing the error (in red) or warning (in yellow) and (iii) by showing the error message as a tooltip when the user hovers

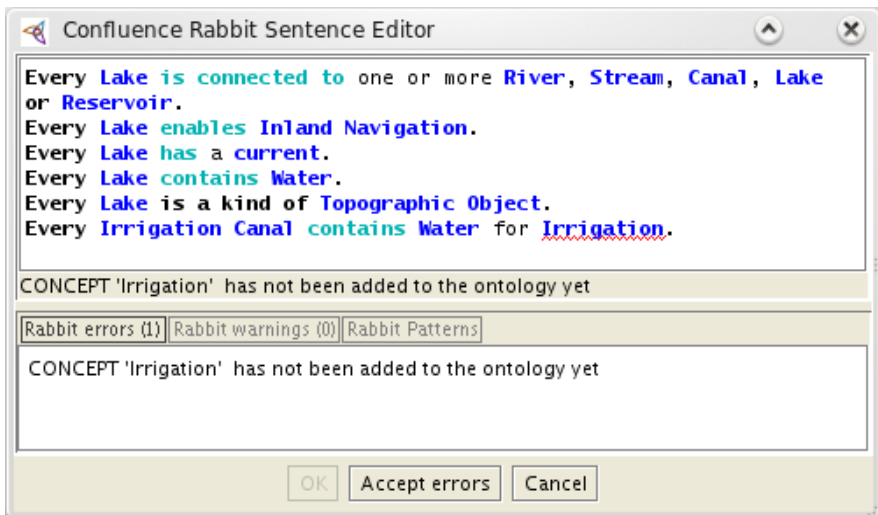


Fig. 4. Rabbit Editor component showing syntax highlighting and error feedback

over the sentence with the mouse. Fig. 4 show an screenshot of the Rabbit editor that shows syntax highlighting and error feedback.

Finally, the editor also has a list of sentence structures to remind the user of the correct Rabbit syntax. This list is used in combination with the help documentation described above.

6 Evaluation

Continuous user studies are performed at Ordnance Survey to examine the usability and the use of the Rabbit language (without tool support) to define sample domain ontologies [4,18]. The user interface components of ROO have been evaluated during the development of the tool. A more comprehensive evaluation of ROO has been conducted at the University of Leeds with 16 volunteers from the departments of Geography (8 students) and Earth and Environment (8 students)[21]. During this study domain experts were asked to perform ontology modeling tasks similar to those that would be expected of domain experts by the Ordnance Survey. To examine the benefits of the support offered in ROO, the interaction with ROO was compared to another authoring tool – ACE View [10] – which provides similar interaction means. Here we only present statistically significant and relevant conclusions of the study. A detailed description of the study setup and results is presented in [21].

The study showed that domain experts were able to perform ontology authoring tasks after only a minimal amount of training. The domain experts in the study only received a training of about 10 minutes during which they could read an introduction to the Rabbit or ACE language that showed example sentences.

Regarding *usability*, users found ACE View slightly frustrating to use, while this was not the case for ROO. ROO was rated as intuitive to use. In particular, the users found the error messages in ACE View slightly confusing, while rating the error messages in ROO as not confusing. At the same time, users of both tools thought that the error messages helped them to write correct sentences. We believe these results are due to ACE View lacking syntax highlighting and showing users more technical error messages (which assume knowledge of OWL or linguistics).

An analysis of the quality of the resulting ontologies shows that both tools resulted in incomplete ontologies that were not fit for purpose. One study-related reason for this is that users only had 1 hour to build their ontologies, which is a short time to build a comprehensive ontology of a domain. Users of both tools produced ontologies with modelling problems such as multiple tangled inheritance. Some of these modelling problems were much more common in ACE View than in ROO, such as the creation of OWL instances instead of OWL classes. This problem is common in ACE View because users do not need to declare concepts and individuals but need to use the correct ACE structures. For example sentence **Flood-plain borders a river** represents an instance **Flood-plain**, and should have been written as **Every Flood-plain borders a river** to describe a concept. Although ROO is not totally immune to these

type of problems, they were much less common due to the requirement to explicitly introduce concepts and instances and syntax highlighting of the parsed structure.

Summarising, the evaluation study shows that the mere use of a CNL interface to build ontologies makes it possible for domain experts to start creating ontologies, but is not enough to avoid introducing modelling errors and can be experienced as frustrating. Intelligent tool support that caters for novice users is essential to improve the usability of these tools as shown by the tool support provided by ROO, which makes the process of constructing ontologies using a CNL less error prone for domain experts.

7 Conclusion

In this paper, we presented ROO, a tool that has been designed to cater for the needs of domain experts with little or no ontology engineering experience. We presented the design decisions of the Rabbit language, the ROO tool and the implementation of a parser for Rabbit. In particular, we gave a detailed description of novel aspects of ROO such as the handling of ambiguity by the Rabbit parser, the use of a consistent and easy to understand terminology for error feedback and the use of an ontology creation methodology to support users in the ontology authoring process. Finally, we reviewed evaluation studies that provide empirical evidence in support of using intelligent techniques to assist ontology authors.

Since the first release of ROO, the tool has been downloaded more than 550 times. ROO has been used in the AWESOME project for academic writing and it is being used by PhD students at the University of Leeds. ROO will also be used in a new project in the area of Serious Games [22].

As a next step in this work, we are investigating how to provide better adaptive tool support based on the particular perspective of ontology authors [23], for example by eliciting and formalising the ontology purpose [24].

References

1. Funk, A., Davis, B., Tablan, V., Bontcheva, K., Cunningham, H.: Controlled Language IE Components version 2. SEKT Project deliverable D.2.2.2, University of Sheffield (2007)
2. Kovacs, K., Dolbear, C., Hart, G., Goodwin, J., Mizen, H.: A Methodology for Building Conceptual Domain Ontologies. Technical Report IRI-0002, Ordnance Survey Research Labs (2006)
3. Dolbear, C., Hart, G., Goodwin, J., Zhou, S., Kovacs, K.: The Rabbit Language: description, syntax and conversion to OWL. Technical Report IRI-0004, Ordnance Survey Research Labs (2007)
4. Hart, G., Johnson, M., Dolbear, C.: Rabbit: Developing a control natural language for authoring ontologies. In: Proceedings of the 5th European Semantic Web Conference, pp. 348–360. Springer, Heidelberg (2008)

5. Schwitter, R., Kaljurand, K., Cregan, A., Dolbear, C., Hart, G.: A comparison of three controlled natural languages for OWL 1.1. In: 4th OWL Experiences and Directions Workshop (OWLED 2008), Washington, DC. CEUR Workshop Proceedings, CEUR-WS, vol. 496 (2008)
6. Funk, A., Tablan, V., Bontcheva, K., Cunningham, H., Davis, B., Handschuh, S.: Clone: Controlled language for ontology editing. In: Aberer, K., Choi, K.-S., Noy, N., Allemang, D., Lee, K.-I., Nixon, L.J.B., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudré-Mauroux, P. (eds.) ASWC 2007 and ISWC 2007. LNCS, vol. 4825, pp. 142–155. Springer, Heidelberg (2007)
7. Kaljurand, K., Fuchs, N.: Bidirectional mapping between OWL DL and Attempto Controlled English. In: Alferes, J.J., Bailey, J., May, W., Schwertel, U. (eds.) PP-SWR 2006. LNCS, vol. 4187, pp. 179–189. Springer, Heidelberg (2006)
8. Cregan, A., Schwitter, R., Meyer, T.: Sydney OWL Syntax—towards a controlled natural language syntax for OWL 1.1. In: Proceedings of the OWLED 2007 Workshop on OWL: Experiences and Directions, Innsbruck, Austria, CEUR-WS, vol. 258 (2007)
9. Schwitter, R., Ljungberg, A., Hood, D.: Ecole: A look-ahead editor for a controlled language. In: Proceedings of the Joint Conference combining the 8th International Workshop of the European Association for Machine Translation and the 4th Controlled Language Application Workshop, pp. 141–150 (2003)
10. Kaljurand, K.: ACE View—an ontology and rule editor based on Attempto Controlled English. In: Fifth International workshop on OWL Experiences and Directions (2008)
11. Angele, J., Moench, E., Oppermann, H., Staab, S., Wenke, D.: Ontology-based query and answering in chemistry: Ontonova project halo. In: Proceedings of the 2nd International Semantic Web Conference, pp. 913–928. Springer, Heidelberg (2003)
12. Clark, P., Harrison, P., Murray, W.R., Thompson, J.: Naturalness vs. predictability: A key debate in controlled languages. In: Fuchs, N.E. (ed.) Pre-Proceedings of the Workshop on Controlled Natural Language, CEUR-WS, vol. 448 (2009)
13. Cimiano, P., Haase, P., Heizmann, J., Mantel, M., Studer, R.: Towards portable natural language interfaces to knowledge bases – The case of the ORAKEL system. *Data & Knowledge Engineering* 65(2), 325–354 (2008)
14. Tablan, V., Damjanovic, D., Bontcheva, K.: A natural language query interface to structured information. In: Proceedings of the 5th European Semantic Web Conference, Tenerife, Spain, pp. 361–375. Springer, Heidelberg (2008)
15. Bernstein, A., Kaufmann, E., Fuchs, N., von Bonin, J.: Talking to the semantic web- a controlled english query interface for ontologies. In: 14th Workshop on Information Technology and Systems, pp. 212–217 (2004)
16. Hart, G., Dolbear, C., Goodwin, J.: Lege feliciter: Using structured english to represent a topographic hydrology ontology. In: Proceedings of the 3rd OWL Experiences and Directions Workshop, CEUR-WS, vol. 258 (2007)
17. Horridge, M., Drummond, N., Goodwin, J., Rector, A., Stevens, R., Wang, H.: The manchester OWL syntax. In: Proceedings of the OWLED 2006 Workshop on OWL: Experiences and Directions, CEUR-WS.org, vol. 216 (2006)
18. Engelbrecht, P., Hart, G., Dolbear, C.: Talking rabbit: a user evaluation of sentence production. In: Pre-Proceedings of the Workshop on Controlled Natural Language, vol. 448 (2009)
19. Ordnance Survey's GeoSemantics team: Hydrology ontology (2009), <http://www.ordnancesurvey.co.uk/oswebsite/ontology/Hydrology/v2.0/Hydrology.owl> (last accessed on 21/09/2009)

20. Cunningham, H., Maynard, D., Bontcheva, K., Tablan, V., Ursu, C., Dimitrov, M., Dowman, M., Aswani, N., Roberts, I., Li, Y.: et al.: Developing Language Processing Components with GATE version 5 (a User Guide). User guide, University of Sheffield (2009)
21. Dimitrova, V., Denaux, R., Hart, G., Dolbear, C., Holt, I., Cohn, A.: Involving Domain Experts in Authoring OWL Ontologies. In: International Semantic Web Conference, pp. 1–16. Springer, Heidelberg (2008)
22. Michael, D., Chen, S.: Serious games: Games that educate, train, and inform. Muska & Lipman/Premier-Trade (2005)
23. Denaux, R.: Multi-perspective Ontology Engineering. In: Proceedings of Doctoral Consortium at the 8th International Semantic Web Conference (2009) (to be published)
24. Denaux, R., Cohn, A.G., Dimitrova, V., Hart, G.: Towards Modelling the Intended Purpose of Ontologies: A Case Study in Geography. In: Proceedings of the Terra Cognita Workshop in conjunction with the 8th International Semantic Web Conference, CEUR-WS.org, vol. 518 (2009)

Writing Clinical Practice Guidelines in Controlled Natural Language^{*}

Richard N. Shiffman¹, George Michel¹, Michael Krauthammer¹,
Norbert E. Fuchs², Kaarel Kaljurand², and Tobias Kuhn²

¹ School of Medicine, Yale University
{richard.shiffman,george.michel,michael.krauthammer}@yale.edu
<http://gem.med.yale.edu/ergo/>

² Department of Informatics and Institute of Computational Linguistics,
University of Zurich
{fuchs,kalju,tkuhn}@ifi.uzh.ch
<http://attempto.ifi.uzh.ch>

Abstract. Clinicians could benefit from decision support systems incorporating the knowledge contained in clinical practice guidelines. However, the unstructured form of these guidelines makes them unsuitable for formal representation. To address this challenge we translated a complete set of pediatric guideline recommendations into Attempto Controlled English (ACE). One experienced pediatrician, one physician and a knowledge engineer assessed that a suitably extended version of ACE can accurately and naturally represent the clinical concepts and the proposed actions of the guidelines. Currently, we are developing a systematic and replicable approach to authoring guideline recommendations in ACE.

1 Introduction

Contemporary healthcare is characterized by widespread practice variation and overuse, underuse, and misuse of medical resources [10,11,12]. To address these issues, a worldwide initiative to create, disseminate, and implement clinical practice guidelines has arisen.

Clinical practice guidelines are defined as “systematically developed statements to assist practitioner and patient decisions about appropriate healthcare for specific clinical circumstances” [2]. Modern guidelines help to make explicit the scientific basis for guideline statements. In their most common form, guidelines are documents that contain recommendations that support clinical decision-making by healthcare professionals and patients.

More than 4000 guidelines on hundreds of clinical topics have been published by a wide variety of organizations¹. These guidelines summarize the most current understanding of what constitutes “best practice”.

^{*} This work was supported by the US National Library of Medicine and the Agency for Healthcare Research and Quality (grant R01 LM007199).

¹ <http://www.g-i-n.net>

Guidelines are developed by teams of medical experts who systematically review and appraise the relevant biomedical literature and apply rigorous methods to link recommendations about appropriate care to the supporting scientific evidence. In general, the development process involves: (1) topic selection, (2) convening a multidisciplinary panel, (3) defining scope and questions to be addressed, (4) searching the biomedical literature for relevant evidence and filtering that evidence to ascertain the most representative and valid subset, (5) evaluating evidence quality and creating evidence tables, (7) developing consensus about appropriate care, (8) formulating recommendation statements and assigning a recommending strength, (9) submitting the draft guideline for peer review, and (10) publishing the final product.

However, guideline development is often handicapped by problems in 5 areas that detract from their use and usability: (1) defects in guideline quality, (2) imprecise guideline language, (3) non-transparent knowledge synthesis, (4) ineffective implementation and uptake, (5) problems formalizing guidelines. We will explore each of these areas of deficiency and then describe an approach to addressing these issues that applies controlled natural language tools.

1.1 Defects in Guideline Quality

A decade ago, Grilli [13] and Shaneyfelt [14] pointed out that guidelines were not adhering to emerging quality standards. Criteria that define valid and useful guidelines have been established and codified in the AGREE instrument [15] and in the Conference on Guidelines Standardization (COGS) statement [16]. Grilli reported that after examining 431 specialty society guidelines, 87% did not report whether a literature search had been done, and 67% did not describe the type of professionals involved in guideline development. Such defects in documentation represent a threat to guideline quality. We propose a tool that incorporates reminders to guideline authors about requisite content and prompts them to document relevant information.

1.2 Imprecise Guideline Language

As early as 1995, Tierney et al. [17] noted that “as written, guidelines are often difficult or impossible to implement”. They suggested that authors write recommendations as rules in a simple “If-then-else” format with all the parameters strictly defined. An updated view of this proposal suggests that authors should be explicit about: when (i.e. under what circumstances); who; ought (i.e. at which level of obligation); to do what; to whom; how; why.

Because guideline authors often do not address these questions, guideline language is often vague and underspecified, and sometimes even frankly ambiguous [18]. True ambiguity, i.e. statements that can be interpreted in two or more discrete ways, is rarely intentional. However, vagueness and underspecification are most often introduced when there is insufficient high-quality evidence to support a recommendation. They are also used when the authors are unable to reach consensus, when they have concerns about setting a legal standard of

care, and when economics dictate a course that reflects scarce resources. It would be valuable to implementers if the authors were explicit about the reasons for deliberate vagueness or underspecification.

Experience with the Guidelines Implementability Appraisal indicates that each recommendation must be decidable (i.e., the guideline's intended audience should consistently determine whether each condition in the recommendation has been satisfied), and each recommendation must be executable (i.e. the recommended action should be stated specifically and unambiguously), so that members of the intended audience will execute the recommended action in a consistent way [19]. Yet many current guidelines present statements of fact as "recommendations". For example, a recent guideline on breast cancer management includes the following text as recommendation: "Adjuvant hormone therapy for locally advanced breast cancer results in improved survival in the long term." The statement is not executable as written because it does not indicate under what circumstances, who should do what to whom, how, and with what level of obligation. Such statements of fact cannot be implemented without someone, in many cases the implementer rather than the guideline author, answering these questions.

Additional implementation difficulties are brought about by use of the passive voice, which obscures the actor. Likewise, in recommendations that are aimed at clinicians, statements such as "Patients should receive ..." are unclear about how such an event is to occur. Guidelines should recommend actions that are within the purview of their intended audience.

Another common but troublesome construction is the use of the word "consider" to decrease the level of intended obligation of a verb. It is rarely possible to measure whether an action was "considered". Since guideline recommendations are often intended to be part of quality improvement efforts, the lack of measurability is problematic.

1.3 Deficiencies in Knowledge Synthesis

Guideline authors should explicitly define the quality of evidence that supports recommendation statements and assign a level of recommendation strength. Evidence quality is an indication of the authors' confidence in their appraisal of what benefits and harms can be anticipated if the recommendation is followed [20]. It is based on an analysis of the validity, consistency, and relevance of the scientific evidence supporting a recommendation statement. For example, multiple, well-conducted randomized controlled trials on populations similar to a guideline's target population provide higher confidence than observational studies (cohort and case-control studies). Likewise observational studies provide higher "quality" evidence than case-series and expert consensus. Recommendation strength communicates the authors' assessment of the importance of adhering to the recommendations. It is based on a value judgment about anticipated benefits, risks, harms, and costs associated with adhering to a recommendation, as well as a consideration of evidence quality. Recommendation strength is particularly important to guideline implementers who must design systems that support

adherence. Hussain found that less than 41% of randomly selected guideline statements were accompanied by an indicator of recommendation strength [21].

1.4 Ineffective Implementation and Uptake

Balas calculated that only 14% of research findings filter down to everyday practice and that it takes an average of 17 years to do so [22]. This lag in the incorporation of scientific advances into clinical care creates delays that deprive patients of potential health benefits [23]. New evidence alone rarely leads to improvements in practice [24,25]. Effective mechanisms are needed to influence clinicians to adopt practices based on evidence when such behavior change is justified.

Practice guidelines constitute an important mechanism that can reduce the delivery of inappropriate care and support the introduction of new knowledge into clinical practice [26]. But the knowledge that they contain must be delivered in an effective manner.

Grimshaw and Russell showed that the highest probability of influencing clinician behavior occurs when patient-specific reminders are delivered at the time and place of a consultation [27]. An electronic clinical decision support system (CDSS) is a system that compares patient characteristics with a knowledge base and then guides a health provider by offering patient-specific and situation-specific advice.

1.5 Problems Formalizing Guidelines

There is a mismatch between the unstructured narrative form of published guidelines and the formality that is necessary for the operationalization of guideline knowledge in clinical decision support systems. Uncritical translation of recommendations into computable statements risks distortion of the guideline authors intent. Patel and Ohno-Machado demonstrated that experience and background of knowledge engineers impacts the accuracy of translation [28,29].

2 Attempto Controlled English (ACE)

In the ERGO Project (Effective Representation of Guidelines with Ontologies)² we will demonstrate the feasibility of translating guideline knowledge into rules. We propose to use Attempto Controlled English (ACE) [3] as an intermediate representation between the implicit knowledge contained in the minds of the domain experts and the representation of that knowledge in an explicit computable form.

ACE is a controlled natural language, i.e. a precisely defined subset of English with restrictions on vocabulary and grammar. These restrictions result in increased terminological consistency, reduced ambiguity, consistent vocabulary,

² <http://gem.med.yale.edu/ergo/>

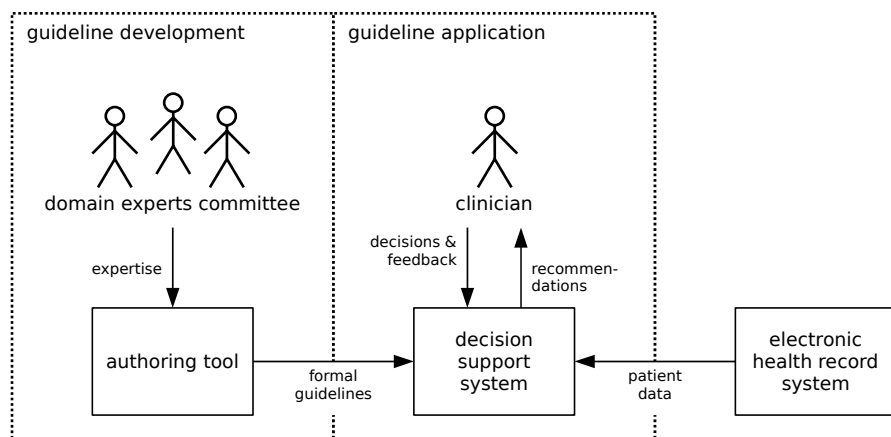


Fig. 1. This diagram shows the general architecture of our approach. It shows the information and data flows between systems (in rectangular shapes) and users. The current focus of the ERGO project lies on the guideline development part.

potentially templated phrases, and a generally simplified sentence and text structure. ACE has the additional benefit of being supported by a parsing engine that translates ACE texts into first-order logic, thus providing a computable format and supporting automatic reasoning. We plan to use ACE to encode the summary recommendation statements that form the backbone of guideline documents. Often published in boldface, these key action statements embody the critical knowledge about appropriate practice that is amplified by supporting text.

Our primary goal is to develop an authoring tool that helps guideline authors to reduce ambiguity, vagueness, incompleteness, and inconsistency, and facilitates the translation of guideline recommendations into logic statements that can be implemented in decision support systems. These systems generally depend on production rules derived from guideline recommendations to create a knowledge base. The decision support system compares an individual patient's characteristics (demographic descriptors and clinical findings) against these rules to guide a health provider by offering patient-specific and situation-specific advice. A second goal is to demonstrate that ACE is a good candidate controlled natural language for writing recommendations. Figure 1 shows the general architecture of our approach.

3 Methods

A critical first step is to establish whether clinical guidelines can be adequately expressed in ACE, and to identify potential barriers to the effective translation of natural language guideline recommendations into controlled natural language. To answer this question we decided to manually “ACE”ify the set of recommendations

contained in the guideline “Diagnosis, Treatment, and Evaluation of the Initial Urinary Tract Infection in Febrile Infants and Young Children” (UTI) [1].

UTI was chosen because (1) it includes a sufficient number of recommendations to exercise the translation process, (2) its recommendations involve a variety of action types and levels of obligation, (3) some recommendations incorporate a temporal sequence, (4) while others contain sentences related by anaphoric references. In spite of its brevity, this guideline demonstrates many challenges in translating recommendations.

The recommendations were translated by 3 members of the ACE team at the University of Zurich (NEF, TK, KK). They individually created ACE statements for each of the key action statements in the published guideline. Since the goal was to determine feasibility, not reliability, we encouraged independent analysis and translation.

Once translated into ACE, the recommendations were reviewed by a pediatrician with expertise in clinical guidelines (RNS), another physician (MK), and by a knowledge engineer (GM). Judgments were made regarding the accuracy of translation and the naturalness of the ACE statements. Obstacles encountered in the translation process were captured and discussed after all translations were completed.

4 Results

All eleven UTI key action statements were successfully translated into ACE (cf. Appendix). The controlled natural language experts approached the translation problem somewhat differently, but each was able to create rules from each natural language recommendation statement. The rules that they created are interrelated, i.e. they have preconditions that must be fulfilled by other rules, and consequences that can trigger other rules.

As might be expected, there was some level of variability in the ACE translations performed by the 3 ACE experts working independently. Some of the textual variation is of course due to the fact that each rule can only be fully understood in the context of the other rules.

For example, the guideline natural language statement: “Infants and young children 2 months to 2 years of age, including those whose treatment initially was administered parenterally, should complete a 7- to 14-day antimicrobial course orally (Strength of Evidence-Strong).” was transformed into the following statements.

NEF: Every young child must complete an oral antimicrobial-therapy for at least 7 days and at most 14 days.

TK: If the patient is a young child who has a UTI then the doctor must administer an oral antimicrobial-therapy that lasts at least 7 days and that lasts at most 14 days.

KK: Every febrile-young-child that has a UTI must undergo an antimicrobial-therapy that is oral and that lasts at least 7 days and that lasts no more than 14 days.

Table 1. A comparison of three of the eleven original natural language guidelines together with their ACE equivalents

Original guideline	Attempto Controlled English
The presence of UTI should be considered in infants and young children 2 months to 2 years of age with unexplained fever (strength of evidence: strong).	If the patient is a young child who has an unexplained fever then the clinician must consider UTI.
In infants and young children 2 months to 2 years of age with unexplained fever, the degree of toxicity, dehydration, and ability to retain oral intake must be carefully assessed (strength of evidence: strong).	If the patient is a young child who has an unexplained fever then the clinician must assess the degree of Toxicity and must assess the degree of Dehydration and must assess the Ability-to-retain-oral-intake.
If an infant or young child 2 months to 2 years of age with unexplained fever is assessed as being sufficiently ill to warrant immediate antimicrobial therapy, a urine specimen should be obtained by SPA or transurethral bladder catheterization; [...] (strength of evidence: good).	If the patient is a young child who has an unexplained fever and the patient is sufficiently-ill then the clinician should analyze a culture of a urine-specimen that is obtained-by SPA or that is obtained-by Transurethral-catheterization.

Table 1 shows three of the original natural language guidelines together with their ACE equivalents.

The pediatrician, the physician and the knowledge engineer concluded that ACE is capable of accurately stating the clinical concepts and the actions described in the guidelines recommendations. ACE statements were judged to be acceptably “natural” sounding by the pediatrician (RNS), a native English speaker.

We interpret the ACE guidelines as forward-chained rules that are executed under the control of the clinician. Every rule consists of preconditions that must be fulfilled to trigger the rule, and conclusions that are true after the rule fired, and that can be used as preconditions for other rules. Execution under the control of the clinician means that after each execution of a rule the clinician decides whether he/she is satisfied with the conclusions found so far, or to execute further rules.

In addition to the guidelines, the experts created a UTI background ontology that included such statements as:

- Every child is a person.
- SPA is a method.
- No analysis confirms X and excludes X.
- Every antimicrobial-therapy is a therapy.
- ...

To get the rule machinery running, a number of initial facts are asserted that originate from the patients electronic health record or that are manually asserted by the clinician, for instance:

The patient is a young child.
 The patient's age is 1.5 years.
 The patient has an unexplained fever.
 ...

The complete ACE version of the UTI guidelines is found in the appendix.

5 Obstacles to Translation

Several obstacles were encountered in the course of translation. The means by which each was addressed is described below.

Medical Terminology. The ACE vocabulary comprises predefined function words (e.g., determiners, conjunctions, prepositions) and about 100,000 content words (nouns, verbs, adjectives, adverbs). However, specialized medical terminology is not part of the ACE lexicon. Several large standardized vocabularies of medical terms exist, such as UMLS, SNOMED and LOINC. ACE has “hooks” by which external vocabularies can be incorporated. In future work we plan to examine the feasibility of incorporating components of these vocabularies.

Though many of the medical terms can be found in the above mentioned lexicons, the problem remains that terms, such as “ability to retain oral intake”, “sufficiently ill” and “SPA”, require clear and consistent specifications by guideline authors. We plan to solve this problem by providing an authoring tool that accepts only terms that are known to the system and that have a clearly defined meaning.

It is possible to temporarily add content words to ACE by prefixing unknown words with their respective word-class, for example, v:reevaluate, n:imaging-studies, and a:ill-appearing. Using this approach, it was possible to translate unrecognized medical terms into ACE constructs without using a specialized lexicon.

Level of Obligation. Considerable uncertainty accompanies most medical decision making. Variation in evidence validity as well as the accuracy of clinical observations and measurements contribute to this uncertainty.

Guideline recommendations imply a level of obligation. This level is conveyed in two different ways. Specific statements that codify the quality of evidence supporting the recommendation and/or the “strength” of the recommendation accompany many guideline recommendations. In addition, guideline recommendation statements often contain deontic terminology indicated by modal auxiliaries (e.g., “must”, “should”, “may”) or by use of other constructions (e.g., “is appropriate”, “the Committee strongly recommends”).

At the outset of this study, only the modals “can” and “must” were available in ACE. Lomotan and colleagues demonstrated in [30] that these are not sufficient to capture the range of obligations imposed by recommendations. It has already been noted as a Best Practice ³ that a limited vocabulary of “must/required/shall”,

³ <http://www.faqs.org/rfcs/rfc2119.html>

“should/recommended”, and “may/optional” (and their negations) represent a limited vocabulary of keywords for use in Internet Requests for Comments. In guideline recommendations “should” is the most frequently used modal with a level of obligation between “can” and “must”. To adequately represent the required levels, ACE was extended by the modals “should/it is recommended that” and “may/it is admissible that”, and their negated forms. This is already reflected in the examples of table 1.

6 Discussion and Future Work

In this work, we demonstrated that ACE can be used to express the recommendation statements contained in clinical practice guidelines. These statements express the semantics of the natural language in a format that is suitable for standardization. ACE guideline recommendation statements are natural sounding, yet may be constrained in terms of grammar, vocabulary and style. Furthermore, ACE statements can be translated to discourse representations structures in an automated manner that can be further transformed into computer-interpretable statements.

Now that we have demonstrated feasibility, our immediate plan is to build a “look-ahead” editor for clinical practice guidelines expressed in ACE. This editor is intended for use by guideline authors to remedy the defects in the development process described in the introduction. The editor will incorporate 4 modules:

To improve the comprehensiveness of documentation of the elements necessary to establish guideline validity and facilitate usability, we plan to include a module that prompts authors to include each of the COGS checklist components. In many cases, a guideline authoring group will be able to store standard language that is repeated in each guideline, e.g., how conflicts of interest are handled and the scheme for encoding evidence quality.

To facilitate knowledge synthesis, a module, perhaps configured as a wizard, will lead the developers through a process of recording aggregate evidence quality that supports each recommendation, the benefits, risks, harms, and costs anticipated if the recommendation is carried out, and the judgment of the authors regarding whether there is a preponderance of benefit or harm or an equilibrium between them. That information will be used to define a recommendation strength for each statement. The recommendation strength will dictate and constrain choices for deontic terminology. Strong recommendations will permit use of “must”, whereas lower level statements will limit the authors to the use of “should” or “may”.

We plan to incorporate a WYSIWYM editor that dynamically displays the knowledge defined so far and the specific options available for extending or revising it. Such an editor module (similar to the existing ACE Editor ⁴) can be used to address the imprecise language that is common in current guideline statements. This approach was described by Scott et al. [6], and has been used by Schwitter [7] and by Kuhn [5] working with controlled language grammars. The editor will constrain authors to write in Attempto Controlled English and can

⁴ <http://attempto.ifi.uzh.ch/aceeditor>

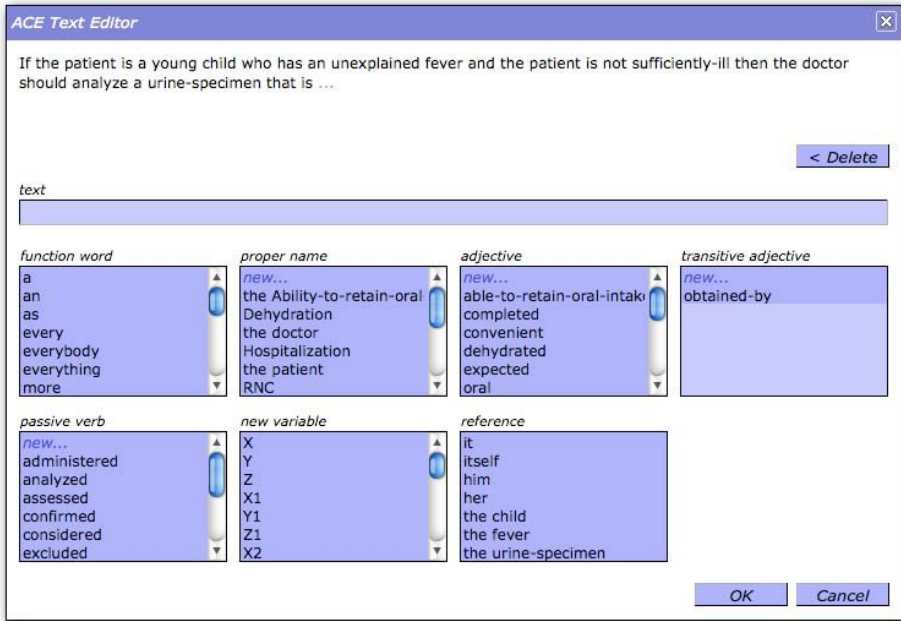


Fig. 2. This screenshot shows Kuhn’s ACE Editor adapted to the UTI terminology

include specific constraints that address identified problems, e.g., flagging the use of the term “consider”. Figure 2 shows a screen-shot of Kuhn’s ACE Editor, adapted to the terminology of UTI. After the author has entered “If the patient is a young child who has unexplained fever and the patient is not sufficiently-ill then the doctor should analyze a urine-specimen that is”, the ACE Editor expects the author to enter the next word from the syntactically possible choices offered by the scroll-menus. Alternatively, the author can freely enter text into the “text” field.

Finally, to diminish the problem of delayed uptake and use of knowledge contained in guidelines, we plan to take advantage of ACE’s ability to facilitate translation into computable formats. We will transform the rules that are defined into a standardized computer language (Arden Syntax [31]), and embed these rules within a computer-mediated decision support system. That system will combine the rules with clinical observations about specific patients that are derived from an electronic health record system to provide guidance about best practices for care for that patient. As is the case with all decision support applications, the advice provided is advisory and is always presented as such to the users.

References

1. American Academy of Pediatrics. Practice Parameter: The Diagnosis, Treatment, and Evaluation of the Initial Urinary Tract Infection in Febrile Infants and Young Children. *Pediatrics* 103, 843–852 (1999)

2. Field, M.J., Lohr, K.N. (eds.): Guidelines for Clinical Practice: From Development to Use. National Academy Press, Washington (1992)
3. Fuchs, N.E., Kaljurand, K., Kuhn, T.: Attempto Controlled English for Knowledge Representation. In: Reasoning Web, 4th International Summer School 2008, Tutorial Lectures (2008)
4. Greenes, R.A., Peleg, M., Boxwala, A., Tu, S., Patel, V., Shortliffe, E.H.: Sharable Computer-Based Clinical Practice Guidelines: Rationale, Obstacles, Approaches, and Prospects. In: Medinfo. (2001)
5. Kuhn, T.: AceWiki: Collaborative Ontology Management in Controlled Natural Language. In: 3rd Semantic Wiki Workshop, CEUR Workshop Proceedings (2008)
6. Power, R., Scott, D., Evans, R.: What You See Is What You Meant: Direct Knowledge Editing with Natural Language Feedback. In: Proc. 13th European Conference on Artificial Intelligence, ECAI 1998 (1998)
7. Schwitter, R., Ljungberg, A., Hood, D.: ECOLE: A Look-ahead Editor for a Controlled Language. In: Proc. 8th European Association for Machine Translation and the 4th Controlled Language Applications Workshop, EAMT-CLAW 2003 (2003)
8. Shiffman, R.N., Michel, G., Essaihi, A., Thornquist, E.: Bridging the Guideline Implementation Gap: A Systematic, Document-centered Approach to Guideline Implementation. *J. Am. Med. Informatics Assoc.* 5, 418–426 (2004)
9. Sun, Y., van Wingerde, F.J., Homer, C.J.: The Challenges of Implementing a Real-Time Clinical Practice Guideline. *Clin. Perform Quality Health Care* (1999)
10. Detsky, A.S.: Regional variation in medical care. *N. Engl. J. Med.* (1995)
11. Wennberg, J.E., Gittelsohn, A.: Small-area variation in health care delivery. *Science* (1973)
12. Wennberg, J.E., Freeman, J.L., Culp, W.J.: Are hospital services rationed in New Haven or over-utilised in Boston. *Lancet* (1987)
13. Grilli, R., Magrini, N., Penna, A., Mura, G., Liberati, A.: Practice guidelines developed by specialty societies: the need for a critical appraisal. *Lancet* (2000)
14. Shaneyfelt, T.M., Mayo-Smith, M.F., Rothwangl, J.: Are guidelines following guidelines? The methodological quality of clinical practice guidelines in the peer-reviewed medical literature. *JAMA* (1999)
15. AGREE Collaboration. AGREE Instrument. AGREE Collaboration (2001)
16. Shiffman, R.N., Shekelle, P., Overhage, J.M., Slutsky, J., Grimshaw, J., Deshpande, A.M.: Standardized reporting of clinical practice guidelines: a proposal from the Conference on Guideline Standardization. *Ann. Intern. Med.* (2003)
17. Tierney, W.M., Overhage, J.M., Takesue, B.Y., Harris, L.E., Murray, M.D., Vargo, D.L.: Computerizing guidelines to improve care and patient outcomes: the example of heart failure. *J. Am. Med. Inform. Assoc.* (1995)
18. Codish, S., Shiffman, R.N.: A model of ambiguity and vagueness in clinical practice guideline recommendations. In: Friedman, C.P. (ed.) Proceedings of the Annual Meeting of the American Medical informatics Association 2005 (2005)
19. Shiffman, R.N., Dixon, J., Brandt, C., Essaihi, A., Hsiao, A., Michel, G.: The GuideLine Implementability Appraisal: development of an instrument to identify obstacles to guideline implementation (2004) (Submitted)
20. Guyatt, G., Gutterman, D., Baumann, M.H., Adrisso-Harris, D., Hylek, E.M., Phillips, B.: Grading strength of recommendations and quality of evidence in clinical guidelines: report from an American College of Chest Physicians task force. *Chest* (2006)
21. Hussain, T., Michel, G., Shiffman, R.N.: The Yale guideline recommendation corpus: a representative sample of the knowledge content of guidelines. *Int. J. Med. Informatics* (2009)

22. Balas, E.R., Boren, S.A.: Managing clinical knowledge for health care improvement. Yearbook of Medical informatics (2000)
23. Oxman, A.D., Thomson, M.A., Davis, D.A., Haynes, R.B.: No magic bullets: a systematic review of 102 trials of interventions to improve medical practice. *Can. Med. Assoc. J.* (1995)
24. Bero, L.A., Grilli, R., Grimshaw, J.M., Harvey, E., Oxman, A.D., Thomson, M.A.: Closing the gap between research and practice: an overview of systematic reviews of interventions to promote the implementation of research findings. The Cochrane Effective Practice and Organization of Care Review Group (1998)
25. Berwick, D.M.: Disseminating innovations in health care. *Jama* (2003)
26. Merritt, T.A., Palmer, D., Bergman, D.A., Shiono, P.H.: Clinical practice guidelines in pediatric and newborn medicine: implications for their use in practice. *Pediatrics* (1997)
27. Grimshaw, J.M., Russell, I.T.: Effect of clinical guidelines on medical practice: a systematic review of rigorous evaluations. *Lancet* (1993)
28. Ohno-Machado, L., Gennari, J.H., Murphy, S.N., Jain, N.L., Tu, S.W., Oliver, D.E.: The guideline interchange format: a model for representing guidelines. *J. Am. Med. Informatics Assoc.* (1998)
29. Patel, V.L., Allen, V.G., Arocha, J.F., Shortliffe, E.H.: Representing clinical guidelines in GLIF: individual and collaborative expertise. *J. Am. Med. Informatics Assoc.* (1998)
30. Lomotan, E.A., Michel, G., Lin, Z.Q., Shiffman, R.N.: How should we write guideline recommendations? Quality and Safety in Health Care (in press)
31. Jenders, R.A., Hripcsak, G., Sidelli, R.V., DuMouchel, W., Zhang, H., Cimino, J.J., Johnson, S.B., Sherman, E.H., Clayton, P.D.: Medical decision support: experience with implementing the Arden Syntax at the Columbia-Presbyterian Medical Center. In: *Proc. Annu Symp. Comput. Appl. Med. Care* (1995)

A Appendix: Complete ACE Version of UTI Guidelines

The appendix contains the complete ACE text of the UTI guidelines. The rules derived from the UTI guidelines are designed to be executed under the control of the responsible clinician. If the preconditions of a rule hold then the clinician can decide to “fire” the rule. The consequences of this rule then either confirm/exclude a certain diagnosis or suggest — with various levels of obligation — to perform additional tests. In both cases the clinician decides whether or not to execute further rules.

The different levels of strength of evidence are represented in the following way: “strength of evidence: strong” is represented by the modal verb “must”; “strength of evidence: good” is represented by “should”; “strength of evidence: fair” and “strength of evidence: opinion/consensus” are represented by “can”. Note that the modals occur only in the consequences of the rules.

It is assumed that exactly two actors exist: the patient and the doctor. This allows us to simplify the formalization. E.g. instead of “... perform Ultrasonography on the patient”, we can just say “... perform Ultrasonography”, which implicitly and unambiguously refers to the patient.

There is a small background ontology. Furthermore, there are some auxiliary rules (2.2, 2.3, 2.4, 4.3, 4.4, 4.5, 4.6 and 8.2), which do not explicitly occur in the

original guidelines and look somewhat redundant. They are, however, needed to interconnect the rules.

Lines prefixed by “#” are comments that are ignored by the ACE parser.

```
# Background Ontology
# =====

Every child is a person.
Every infant is a person.
Every infant's age is less than 1 year.
Every person whose age is more than 2 months and
    whose age is less than 2 years is a young child.

SPA is a method.
Transurethral-catheterization is a method.

No analysis confirms X and excludes X.

Every antimicrobial-therapy is a therapy.

Ultrasonography is an imaging-study.
VCUG is an imaging-study.
RNC is an imaging-study.

# Background Rules
# =====

# Rule B.1
If
    the doctor administers a therapy
then
    the patient undergoes the therapy.

# Rule B.2
If
    the patient undergoes a therapy
then
    the therapy is completed or
    the therapy is not completed.

# Rule B.3
If
    the doctor performs an imaging-study
then
    the imaging-study is completed or
    the imaging-study is not completed.

# Recommendation 1
# =====

# Rule 1.1
If
    the patient is a young child who has an unexplained fever
then
    the doctor must consider UTI.

# Recommendation 2
# =====

# Rule 2.1
If
    the patient is a young child who has an unexplained fever
then
    the doctor must assess the degree of Toxicity and
```

must assess the degree of Dehydration and
 must assess the Ability-to-retain-oral-intake.

Rule 2.2

If
 the doctor assesses the degree of Toxicity
 then
 the patient is toxic or
 is not toxic.

Rule 2.3

If
 the doctor assesses the degree of Dehydration
 then
 the patient is dehydrated or
 is not dehydrated.

Rule 2.4

If
 the doctor assesses the Ability-to-retain-oral-intake
 then
 the patient is able-to-retain-oral-intake or
 is not able-to-retain-oral-intake.

Recommendation 3

=====

Rule 3.1

If
 the patient is a young child who has an unexplained fever and
 the patient is sufficiently-ill
 then
 the doctor should analyze a culture of a urine-specimen
 that is obtained-by SPA or
 that is obtained-by Transurethral-catheterization.

Recommendation 4

=====

Rule 4.1

If
 the patient is a young child who has an unexplained fever and
 the patient is not sufficiently-ill
 then
 the doctor should analyze a culture of a urine-specimen
 that is obtained-by SPA or
 that is obtained-by Transurethral-catheterization or
 that is obtained-by a convenient method.

Rule 4.2

If
 the patient is a young child who has an unexplained fever and
 the patient is not sufficiently-ill and
 the doctor analyzes a culture of a urine-specimen
 that is obtained-by a convenient method and
 the analysis of the culture suggests UTI
 then
 the doctor should analyze a culture of a urine-specimen
 that is obtained-by SPA or
 that is obtained-by Transurethral-catheterization.

Rule 4.3

If
 the doctor analyzes a culture of a urine-specimen
 that is obtained-by SPA or
 that is obtained-by Transurethral-catheterization

```
then
  the analysis of the culture confirms UTI or
    excludes UTI.

# Rule 4.4
If
  the doctor analyzes a culture of a urine-specimen that is obtained-by a convenient method
then
  the analysis of the culture suggests UTI or
    does not suggest UTI.

# Rule 4.5
If
  the analysis of a culture of a urine-specimen confirms UTI
then
  the patient has UTI.

# Rule 4.6
If
  the analysis of a culture of a urine-specimen excludes UTI
then
  the patient does not have UTI.

# Recommendation 5
# =====
#
# Recommendation 5 is integrated into the recommendations 3 and 4.

# Recommendation 6
# =====

# Rule 6.1
If
  the patient is a young child who has an unexplained fever, and
  the patient is toxic or
    is dehydrated or
    is not able-to-retain-oral-intake
then
  the doctor can administer an antimicrobial-therapy and
    can consider Hospitalization.

# Recommendation 7
# =====

# Rule 7.1
If
  the patient is a young child and
  the analysis of a culture of a urine-specimen confirms UTI
then
  the doctor should administer a parenteral antimicrobial-therapy or
    should administer an oral antimicrobial-therapy.

# Recommendation 8
# =====

# Rule 8.1
If
  the patient is a young child who has UTI and
  the patient undergoes an antimicrobial-therapy for 2 days and
    does not show the expected response of the antimicrobial-therapy
then
  the doctor should reevaluate the patient and
    should analyze a culture of a second urine-specimen.
```


Rule 8.2

If

the patient is a young child who has UTI and
 the patient undergoes an antimicrobial-therapy for 2 days

then

the patient shows the expected response of the antimicrobial-therapy or
 does not show the expected response of the antimicrobial-therapy.

Recommendation 9

=====

Rule 9.1

If

the patient is a young child who has UTI

then

the doctor must administer an oral antimicrobial-therapy that lasts at least 7 days and
 that lasts at most 14 days.

Recommendation 10

=====

Rule 10.1

If

the patient is a young child who has UTI and
 the antimicrobial-therapy of the patient is completed and
 the imaging-study of the patient is not completed

then

the doctor should administer a therapeutically-dosed antimicrobial or
 should administer a prophylactically-dosed antimicrobial.

Recommendation 11

=====

Rule 11.1

If

the patient is a young child who has UTI and
 the patient undergoes an antimicrobial-therapy for 2 days and
 does not show the expected response of the antimicrobial-therapy

then

the doctor can perform Ultrasonography promptly, and
 the doctor can perform VCUG or
 can perform RNC.

Rule 11.2

If

the patient is a young child who has UTI and
 the patient undergoes an antimicrobial-therapy for 2 days and
 shows the expected response of the antimicrobial-therapy

then

the doctor can perform Ultrasonography, and
 the doctor can perform VCUG or
 can perform RNC.

On Controlled Natural Languages: Properties and Prospects^{*}

Adam Wyner, Krasimir Angelov, Guntis Barzdins, Danica Damljanovic, Brian Davis,
Norbert Fuchs, Stefan Hoefler, Ken Jones, Kaarel Kaljurand, Tobias Kuhn,
Martin Luts, Jonathan Pool, Mike Rosner, Rolf Schwitter, and John Sowa

Multiple Institutes

Abstract. This collaborative report highlights the properties and prospects of Controlled Natural Languages (CNLs). The report poses a range of questions concerning the goals of the CNL, the design, the linguistic aspects, the relationships and evaluation of CNLs, and the application tools. In posing the questions, the report attempts to structure the field of CNLs and to encourage further systematic discussion by researchers and developers.

1 Introduction

Controlled Natural Languages (CNLs) are engineered languages which use a selection of the vocabulary, morphological forms, grammatical constructions, semantic interpretations, and pragmatics which are found in a natural language such as English. To date, more than 40 CNLs have been defined, covering English, Esperanto, French, German, Greek, Japanese, Mandarin, Spanish, and Swedish [1]. They facilitate human-human communication (e.g. translation [2] or technical documentation [3]) and human-machine communication (e.g. interfaces with databases [4] or automated inference engines [5]). For example, in the case of human-machine communication, CNLs enable casual users to use formal query languages such as SQL or SPARQL. Given that CNLs are derived from a natural language, a wider range of speakers of the language find it easier to learn, use, read, and write in comparison to other engineered languages such as computer programming languages, formal languages, or international auxiliary languages [6]. It is an exciting time to work on CNLs, which are becoming more sophisticated, useful, and widespread in both academic research and business applications in a range of areas such as aerospace, manufacturing, oil exploration, business rules, public administration, medicine, and biology. CNLs appear to be particularly significant with respect to information extraction of and reasoning with the content of documents on the internet. Indeed, we are ever closer to prototype systems which realise Leibniz's ambition:

^{*} Adam Wyner is the main author, editor, and contact person. Correspondence: Adam Wyner, Department of Computer Science, University College London, Gower Street, London WC1E 6BT, United Kingdom, adam@wyner.info. This report is the outcome of a collaboration amongst the listed contributors; see Section 4 for further contact information. It is based on an original collaborative document:

http://docs.google.com/Doc?id=dd3zb82w_03976bbfm.

The only way to rectify our reasonings is to make them as tangible as those of the Mathematicians, so that we can find our error at a glance, and when there are disputes among persons, we can simply say: Let us calculate, without further ado, to see who is right. [7, p. 51]

In this short report, which follows from the Workshop on Controlled Natural Languages CNL 2009 (8-10 June 2009, Marettimo Island, Italy)¹ and subsequent community discussion, we outline some of the key *properties* that motivate and guide the design of particular CNLs as well as relate one CNL to another. We highlight key areas for future development and deployment. In this respect, our report draws attention to the interrelations and cohesion among CNLs in contrast to [8], which emphasises the divergences. This report is intended to stimulate further discussion among developers of CNLs and development of the languages.

Our intention is not to define CNLs in general (necessary and sufficient conditions), or to distinguish them from sublanguages or language fragments, or to classify current CNLs. The report is also not a review of the extensive literature (see the links and references on Controlled Natural Languages).² By the same token, we have not strictly distinguished “natural” from “engineered” languages, a distinction which relies on assumptions: reading and writing are not “natural” since they are highly engineered and require explicit instruction; first language acquisition cannot occur in the absence of linguistic interaction with speakers of the language; and there is a spectrum of natural languages which are related to a “dominant” language such as creoles, dialects, sets of search terms, early child language, sign language, and so on. Furthermore, within linguistics, there are significant debates concerning, for example, the definition of “word” [9], syntactic grammaticality judgements, and the semantic or pragmatic interpretations of a given sentence [10]. Rather than engage in these issues, our view is that the relations among CNLs arise from a range of properties (some of which have degrees) which can be used to identify a particular CNL in a multidimensional conceptual grid, when compared with other CNLs; in this, one might see specification of the properties and relations as a move towards an *ontology* of engineered languages.

In the following, we first discuss a range of properties of CNLs, then several topics for future development, followed by a link to a wiki which provides a forum for further collaborative discussions on the state of the art and for future developments.

2 The Properties

In this section, we present questions about properties of CNLs. To answer a question is to ascribe a property to the CNL under discussion; we can then enumerate the properties of the language. With these properties, we can relate one CNL to another or relate the CNL to the natural language from which it is derived.

We have distinguished three broad sections from more generic, to design guidelines, and then to more specifically linguistic properties. We do not presume to have fully determined all the properties and the ways they vary; it is for future work to spell this out in detail.

¹ <http://attempto.ifi.uzh.ch/site/cnl2009>

² <http://sites.google.com/site/controllednaturallanguage/>

2.1 Generic Properties

The generic properties relate to high level aspects of the language. While there are overlaps, they are different ways to draw out the specification of the language.

- Who are the intended users? For example, there might be a narrow user base with a specialised application (say cancer researchers) or a broad user base with a generalised application (newspaper comment support).
- What are the purposes? Among other purposes, we may have simplification, standardisation, accessibility, input support, specification of procedural information, information extraction, translation, generation, and inference. Purposes might be tied to specific tasks such as knowledge acquisition, ontology authoring, querying. Depending on the purpose, the language may or may not be translated into a logic formalism or programming language.
- Is the language domain dependent or independent?

The users, purposes, and domain of a CNL determine the requirements which are the design properties that the CNL needs to meet. We discuss requirements in the next subsection.

2.2 Design Properties

The design properties present more specific indicators of the requirements. For several of the properties, usability tests may be conducted as the language is iteratively developed; the results of the tests may be used to guide each subsequent iteration towards the goal of the design of the language. Answers to several of the questions are contingent on the users, purposes, and requirements as well as answers to other design questions. Several of the questions relate to a notion of *habitability* [11]; according to [12] and [13], a language is habitable if: 1) users are able to construct expressions of the language that they have not previously encountered, without significant conscious effort; and 2) users are able to avoid easily constructing expressions that fall outside the bounds of the language.

- Is the language easy to describe, teach, and learn? For example, a CNL is very easy to describe (and thus to teach and learn) if its restrictions, with respect to the full language, can be defined by just a few sentences.
- Is the language easy to read and scan? For example, perhaps a construct such as “not at least” should be disallowed because it is hard to read and better alternatives exist (“less than”). On the other hand, a construct such as “a man that a woman that a dog sees likes waits.” might be allowed, even if it is hard to read because to bar it would violate the principle of compositionality. On the other hand, one might use psycholinguistic heuristics to guide the selection of constructions.
- Is the language easy to write? Is there “syntactic sugar”, which are expressions that are easier to read and write. It might be easier to be able to write: “All different: Paris, London, Tallinn, ...” to express that Paris, London, etc. must be different instances in the model that the sentence describes. Yet it introduces a new keyword “All different” which does not add any semantic expressivity, because the meaning

of the sentence could also be expressed as “Paris is not London. Paris is not Tallinn, ...”. Furthermore, it is debatable whether “All different: Paris, London, Tallinn, ...” is a construction often found in corpora of natural language.

- Is the language easy to understand? A CNL should look and feel like a natural language. The meanings of CNL statements should fit the meaning that readers would naturally assign to them. The selection of words and grammatical patterns might follow some guidelines such as found in the international Plain Language movement³. For example, certain complex constructions could be ruled out.
- Is the language predictable and unambiguous, that is, do the constructions have fixed and constant interpretations in an application domain?
- Is the language formally or informally defined? How accessible is the definition, meaning how much expert knowledge is required to understand the definition?
- How are semantic restrictions handled? Are there sortal restrictions with dependent types, database schemas of relational databases, or a logic in an ontology?
- Are statements translated into a logic? If so, how expressive is the logic? Descriptions logics provide subsets of first order logic, some of which are decidable [14]. In natural language semantics, there are claims that second order logics, which would support quantification over properties and relations, are required [15]. Are intensional expressions (temporal, belief, modal operators) translated into a modal logic? If statements are translated into a logic, what is the relationship – one to one (one form has one meaning and vice versa), one to many (ambiguity), many to one (simplification), or many to many. Where a range of interpretations are available, how is a selection made?
- Are discourses translated into a logic? If so, what sorts of discourse phenomena are addressed such as donkey anaphora, temporal anaphora, and reference to abstract objects (e.g. events, facts, or propositions)?
- What are the formal properties of the CNL in terms of expressivity, tractability, and decidability? What are the consequences, in practice and for the user, of the formal properties?
- How is the CNL evaluated? The CNL might be evaluated according the properties it has and how well it satisfies the requirements. For example, speed, coverage, accuracy of translation, user satisfaction, and so on may be properties used to evaluate the language.
- Is there a mapping to some graphical representation, e.g. conceptual graphs?
- Is the design of the language psycholinguistically motivated? For example, if the definition of the language allows two synonymous sentences, but one is shown by psycholinguistic research to take longer to read and to more often give rise to errors, should the language rule out the problematic sentence?
- Is there an explicit statement of the syntactic and semantic theory which underwrites the language?
- Is the CNL easily and systematically extensible (adding lexical, morphological, syntactic, and semantic elements or components)?
- What is the relationship between the parsing process and semantic representation? Is it a pipeline or parallel structure?

In the next subsection, we outline several of the linguistic properties of interest.

³ <http://www.plainlanguagenetwork.org/>

2.3 Linguistic Properties: Lexicon, Morphology, Syntax, Semantics, and Pragmatics

In this section, we consider a range of linguistic properties that are relevant to the definition of a CNL. As there is great variety, we only consider general points. The choices here may depend, to a great extent, on answers to the design properties. The range of sources for “benchmark” linguistic properties varies greatly: one might take the advanced Oxford Learner’s vocabulary and grammar as the basis; alternatively, the Oxford English Dictionary and a comprehensive grammar [16] might suit; or finally, there is extensive research literature such as that on diathesis [17] or pragmatics [18], among many others. Where and on what basis to make the “cut” in the language may follow from generic or design properties as well as linguistically informed choices.

- What corpus (if any) is one using to judge which linguistic forms to include in the language?
- What linguistic literature or theory (if any) is one using to justify the linguistic properties of the language?
- What classes of nouns, verbs, adjectives, adverbs, quantifiers, etc. are supported?
- Does the lexicon support polysemy or only monosemy? How is polysemy resolved relative to context?
- Is the language mono-lingual, or does it support multi-linguality?
- What morphological word formation rules are supported? Generally, we might expect rules for singular and plural nouns, agreement on verbs, tense and aspect. Some languages require gender and case agreement as well. Are there rules for nominalisation and compounding?
- Are interrogative and imperative forms supported? Are they generated from assertions or must they be explicitly written?
- Are idioms and metaphors allowed? For example, *Bill kicked the bucket*.
- Diathesis alternations (passive-actives, middles, ditransitives, causatives, inchoatives which signal the beginning of an action, and others). What inferential patterns are supported? For example, what are the implications between the following pairs of sentences?
 - ia. John loaded the truck with hay.
 - ib. John loaded hay on the truck.
 - iiia. Ann threw the ball to Beth.
 - iiib. Ann threw Beth the ball.
 - iiia. John opened the door.
 - iiib. The door opened.
- Where we have synonymous syntactic forms (outside the scope of diathesis), which should we adopt? How should relationships between them be defined?
 - ia. Every employee that owns a car is rich.
 - ib. If an employee owns a car then the employee is rich.
 - iiia. John’s car is fast.
 - iiib. There is a car of John. The car is fast.
- Is there syntactic sugar, i.e. redundant expressions that make some expressions easier to state.

- Can there be discontinuous constituent structures, interruptions, or higher-level speech acts? For example, *Bill, so far as anyone knows, isn't in Africa any longer.*
- What sorts of query, relative clause, and sentence subordination markers are supported?
- What sorts of subordinate clauses are supported?
- Is the semantics compositional? If so, what notion of compositionality is at work?
- What logico-syntactic issues are addressed in the semantics such as opacity, quantifier scope, entailment patterns for different sorts of adjectives and adverbs, inference problems in modal logic, and the syntax-semantics interface, where syntactic structures place constraints on semantic interpretation.
- Is discourse supported? What aspects of anaphora are considered: times, locations, facts, propositions, and definite descriptions?
- Is there support for dialogical argumentation, where parties can make contradictory statements?
- Are the syntax and semantics of the language modular in the sense that there are components that can be added or removed to suit particular purposes while still preserving the functionalities of the remaining components? The modules might be (among others):
 - declarative (FOL) statements: *Some carpenter is an artist.*
 - modal operators such as alethic or deontic statements: *Bill may leave; Parties shall not breach terms of contract, otherwise penalties will be applicable.*
 - procedural statements: *She baked bread and then brought it to granny.*
 - propositional attitudes: *She believes that Bill is happy.*
 - mathematical proof statements: *Let's assume N is even. Then...*
 - rhetorical statements: *Rather than waste time on learning, Charles pursued a lucrative career.*
 - temporal, locative operators: *Bill left yesterday. He had been staying at Jill's house.*
- What sorts of presuppositions are supported?

2.4 Relationships and Evaluation

To this point, we have outlined various properties that CNLs may have. In this section, we consider relationships among CNLs, including how they may be evaluated.

- What are the dependencies between the properties?
- What are the subsumption and similarity relations of existing CNLs?
- Is one CNL (or some of its subcomponents) interoperable with another CNL? For example, can the lexicon of a CNL be transformed automatically into a format so that the lexicon can be used by another CNL? Similarly, can the syntactic parse of one be input to the semantic interpreter of another? Is there a CNL “interchange” language, framework, or reference architecture?
- How modular is the CNL? In other words, is it possible to have a “plug and play” architecture for CNLs?

- Should a CNL be developed for only one target language or is it useful to develop a CNL simultaneously in multiple typologically distinct natural languages, so that incompatibilities between languages are discovered and remedied early. This is particularly crucial for translation.
- How can the relative performance of different CNLs be measured? What are the best practices guidelines? What questionnaires and statistical analyses are used? Is performance measured against a “reference” CNL?
- Is there a common “pool” of use cases that we can use to evaluate a CNL?
- What is the measure of “naturalness” or habitability of a lexicon or grammar? Is it linguistic judgement or, for example, statistical occurrence in a corpus of texts such as the British National Corpus or similar?

2.5 Application Properties

Finally, we have several additional considerations that relate to applications of CNLs.

- Are there automatic consistency checks? Consistency is only applicable to CNLs which do semantic interpretation and inference. Is the input statement a contradiction of a statement already in the knowledge base? Is the input statement a contradiction of a statement which is implied by the knowledge base?
- Are there automatic redundancy checks? In other words, can we check whether the input statement is already in the knowledge base of statements? If the CNL includes a semantics, is the input statement implied by statements already in the knowledge base?
- Where the language is associated with an inference engine, are explanations of the inferences given in the CNL and are they context dependent?
- Is there guidance on style? Is the guidance separate from or built into the editing tools? That is, in the first approach, users are given instructions on the style such as "write short and grammatically simple sentences", "use nouns instead of pronouns", "use determiners", and "use active instead of passive". In the second approach, we have “predictive writing” where the editing tools suggest constructions in keeping with the style.
- What support tools are provided by the CNL? Among the tools, there can be a morphological analyser, parser (syntactic), lineariser (mapping from abstract representation to text), (predictive) editing tools which support the formulation of well-formed sentences and give feedback, editing loops, paraphraser, disambiguator/clarificatory dialogue, semantic interpreter, reasoner, and theorem prover.
- Is there a spoken language interface (for either input or output)?
- How is the language maintained and developed? Is the CNL proprietary or open?

3 A CNL Wiki

This paper is but a short overview of key considerations in the design and development of CNLs. For further collaborative discussions and future developments, see the wiki, where researchers discuss better principles, practices, and design patterns that can be of use to CNL developers/designers⁴.

⁴ cnl.wikia.com

4 Contributor Information

- Krasimir Angelov
Chalmers University of Technology and Göteborg University
krasimir@chalmers.se
- Guntis Barzdins
University of Latvia
Guntis.Barzdins@mii.lu.lv
- Danica Damljanovic
University of Sheffield
d.damljanovic@dcs.shef.ac.uk
- Brian Davis
Digital Enterprise Research Institute
brian.davis@deri.org
- Norbert E. Fuchs
University of Zurich
fuchs@ifi.uzh.ch
- Stefan Hoefler
University of Zurich
hoefler@cl.uzh.ch
- Ken Jones
kennethjone@gmail.com
- Kaarel Kaljurand
University of Zurich
kaljurand@gmail.com
- Tobias Kuhn
University of Zurich
kuhntobias@gmail.com
- Martin Luts
ELIKO Competence Centre
martin.luts@eesti.ee
- Jonathan Pool
Utilika Foundation
pool@utilika.org
- Mike Rosner
University of Malta
mike.rosner@um.edu.mt
- Rolf Schwitter
Macquarie University
Rolf.Schwitter@mq.edu.au
- John Sowa
sowa@bestweb.net
- Adam Wyner
University College London
adam@wyner.info

References

1. Pool, J.: Can controlled languages scale to the web? In: Proceedings of the 5th International Workshop on Controlled Language Applications (2006)
2. Mitamura, T., Nyberg, E.: Controlled english for knowledge-based mt: Experience with the kant system. In: Proceedings of the 6th International Conference on Theoretical and Methodological Issues in Machine Translation (TMI 1995), Belgium, Centre for Computational Linguistics, Katholieke Universiteit Leuven, pp. 158–172 (1995)
3. Adriaens, G., Schreors, D.: From cogram to alcogram: Toward a controlled english grammar checker. In: Proceedings of the 14th Conference on Computational Linguistics, vol. 2, pp. 595–601. Association for Computational Linguistics (1992)
4. Mueckstein, E.M.: Controlled natural language interfaces: the best of three worlds. In: CSC 1985: Proceedings of the 1985 ACM thirteenth annual conference on Computer Science, pp. 176–178. Association for Computing Machinery, New York (1985)
5. Bringsjord, S., Arkoudas, K., Clark, M., Shilliday, A., Taylor, J., Schimanski, B., Yang, Y.: Reporting on some logic-based machine reading research. In: Etzioni, O. (ed.) Machine Reading - Papers from the 2007 AAAI Spring Symposium. Number SS-07-06, pp. 23–28. Association for the Advancement of Artificial Intelligence (AAAI Press), Menlo Park (2007)
6. Kuhn, T.: Controlled English for Knowledge Representation. PhD thesis, Faculty of Economics, Business Administration and Information Technology of the University of Zurich (2010)
7. Leibniz, G.: Leibniz: Selections. Charles Scribner's Sons (1951)
8. O'Brien, S.: Controlling controlled english - an analysis of several controlled language rule sets. In: Controlled Translation - Proceedings of the Joint Conference combining the 8th International Workshop of the European Association for Machine Translation and the 4th Controlled Language Application Workshop (EAMT-CLAW 2003), Dublin City University, Ireland, pp. 105–114 (2003)
9. di Sciullo, A.M., Williams, E.: On the definition of Word. MIT Press, Cambridge (1987)
10. Schütze, C.: The Empirical Base of Linguistics. Chicago University Press, Chicago (1996)
11. Watt, W.C.: Habitability. *Amercan Documentation* 19(3), 338–351 (1968)
12. Epstein, S.: Transportable natural language processing through simplicity - the pre system. *ACM Transactions on Information Systems* 3(2), 107–120 (1985)
13. Ogden, W., Bernick, P.: Using Natural Language Interfaces. In: Handbook of Human-Computer Interaction. Elsevier Science Publishers B.V., Amsterdam (1996)
14. Antoniou, G., Assmann, U., Baroglio, C., Decker, S., Henze, N., Patranjan, P.L., Tolsdorf, R. (eds.): Reasoning Web. LNCS, vol. 4636. Springer, Heidelberg (2007)
15. Barwise, J., Cooper, R.: Generalized quantifiers and natural language. *Linguistics and Philosophy* 4, 159–219 (1981)
16. Quirk, R., Greenbaum, S., Leech, G., Svartvik, J.: A Comprehensive Grammar of the English Language. Pearson Longman, London (1985)
17. Levin, B.: English Verb Classes and Alternations: A Preliminary Investigation. University of Chicago Press, Chicago (1993)
18. Kadmon, N.: Formal Pragmatics: Semantics, Pragmatics, Presupposition, and Focus. Wiley Blackwell (2001)

Author Index

- Anderson Healy, Keri 155
Angelov, Krasimir 82, 281
- Bao, Jie 206
Barzdins, Guntis 102, 281
Bos, Johan 121
Braines, Dave 206
- Calvanese, Diego 135
Clark, Peter 65
Cohn, Anthony G. 246
Cramer, Marcos 170
Cunningham, Hamish 187
- Damljanovic, Danica 281
Dantuluri, Pradeep 187
Davis, Brian 187, 281
Denaux, Ronald 246
Dimitrova, Vania 246
Dolbear, Catherine 56, 246
Dragan, Laura 187
- Engelbrecht, Paula 56
- Fisseni, Bernhard 170
Fuchs, Norbert E. 265, 281
- Gruzitis, Normunds 102
- Handschuh, Siegfried 187
Harrison, Phil 65
Hart, Glen 56, 246
Hoefler, Stefan 281
- Jones, Ken 281
- Kaljurand, Kaarel 265, 281
Koepke, Peter 170
Krauthammer, Michael 265
Kühlwein, Daniel 170
Kuhn, Tobias 1, 265, 281
- Luts, Martin 281
- Michel, George 265
Murray, William R. 65
- Pace, Gordon J. 226
Pool, Jonathan 281
Potter, Andrew 21
- Ranta, Aarne 82
Rosner, Michael 226
Rosner, Mike 281
- Schröder, Bernhard 170
Schwitter, Rolf 36, 281
Shadbolt, Nigel R. 206
Shiffman, Richard N. 265
Smart, Paul R. 206
Sowa, John 281
Spreeuwenberg, Silvie 155
- Thompson, John 65
Thorne, Camilo 135
- Veldman, Jip 170
- Wyner, Adam 281