Lecture Notes in Artificial Intelligence 2892

Edited by J. G. Carbonell and J. Siekmann

Subseries of Lecture Notes in Computer Science

Springer
Berlin
Heidelberg
New York
Hong Kong
London
Milan
Paris
Tokyo

Frithjof Dau

The Logic System of Concept Graphs with Negation

And Its Relationship to Predicate Logic

Springer

Author

Frithjof Dau
Technische Universität Darmstadt
Fachbereich Mathematik
Schlossgartenstr. 7
64289 Darmstadt
Germany
E-mail: dau@mathematik.tu-darmstadt.de

Cataloging-in-Publication Data applied for

A catalog record for this book is available from the Library of Congress.

Bibliographic information published by Die Deutsche Bibliothek
Die Deutsche Bibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data is available in the Internet at <http://dnb.ddb.de>.

CR Subject Classification (1998): I.2, F.4.1, G.2.2

ISSN 0302-9743
ISBN 3-540-20607-8 Springer-Verlag Berlin Heidelberg New York

Springer-Verlag is a part of Springer Science+Business Media

springeronline.com

Typesetting: Camera-ready by author, data conversion by Boller Mediendesign
Printed on acid-free paper SPIN: 10966822 06/3142 5 4 3 2 1 0

Preface

All necessary reasoning is strictly speaking mathematic reasoning, that is to say, it is performed by observing something equivalent to a mathematical diagram.

Charles Sanders Peirce

The quotation of Peirce is taken from his 1898 Cambridge lectures: Reasoning and the Logic of Things. Today – more than 100 years later – this title is still of striking topicality. An enormous variety of scientists, from fields such as philosophy, mathematics, computer science, linguistics and biology, devote themselves to research on human knowledge and thinking. Logic, however, which was understood as the theory of the forms of thinking in centuries past, is nowadays understood as *mathematical* logic in a much narrower sense, and has lost its former connection to the so-called *elementary* logic to a large degree.

In his Contextual Logic, Rudolf Wille makes an attempt to take the former understanding of logic into consideration again and to endow it with the same mathematical preciseness that marks mathematical logic. As he says in [77], '*Contextual Logic*' is a kind of logic which is '*grounded on the traditional philosophical understanding of logic as the doctrine of the forms of thinking*' and aims to support knowledge representation and knowledge processing.

Since the 16th century, human knowledge and thinking have been understood to be made up of three parts: concepts, judgements, and conclusions. Concepts are the basic units of thinking, judgements are combinations of concepts and facts, and conclusions are entailments between judgements. Thus, elementary logic was presented in three parts as well, namely the doctrine of concepts, the doctrine of judgements, and the doctrine of conclusions.

Contextual logic is based on a mathematization of these doctrines. Thus, Contextual logic has to provide mathematizations for concepts, judgements and conclusions. Concepts have already been formalized in Formal Concept Analysis (FCA). They are units of thought which are described by their extension and intension. These two components are formalized in FCA. A formal concept is a pair of sets: its extension, which is a set of (formal) objects and is called the *extent of the formal concept*, and its intension, which is a set of (formal) attributes and is called the *intent of the formal concept.* The relationship between the extent and the intent of a concept is as follows: In the intent of a formal concept, we find exactly all attributes which apply to all objects of the extent, and, vice versa, in the extent of the formal concept we find exactly all objects which satisfy all attributes of the intent (for the mathematical details we refer the reader to [19]). FCA was introduced by Wille in 1981, and since then it has been used successfully in more than 200 projects.

An approach for a formalization of judgements and conclusions can be found in Sowa's theory of conceptual graphs (see [59], [65]). Conceptual graphs are based on the existential graphs of Peirce and are developed to express meaning that is humanly readable and understandable as well as precise and computationally tractable. Conceptual graphs can be understood as formal judgements that are closely related to natural language. In particular, they are graphs that consist of concept nodes, which bear references as well as types of the references. The concept boxes are connected by edges, which are used to express different relationships between the referents of the attached concept boxes. Sowa provides rules for formal deduction procedures on conceptual graphs; hence the system of conceptual graphs offers a formalization of conclusions too.

As FCA provides a formalization of concepts, and as conceptual graphs offer a formalization of judgements and conclusions, a convincing idea is to combine these approaches to gain a unified formal theory for concepts, judgements and conclusions, i.e., a formal theory of elementary logic. In [75], Wille marked the starting point for a such a theory. There he provided a mathematization of conceptual graphs where the types of conceptual graphs are interpreted by formal concepts of a so-called power context family. The resulting graphs are called *concept graphs.* They form the mathematical basis for contextual logic.

In her Ph.D.-thesis 'Kontextuelle Urteilslogik mit Begriffsgraphen' ('Contextual Judgement Logic with Concept Graphs,' see [44]) Prediger worked out the theory of concept graphs using the foundation of Wille's ideas. In contrast to Wille, she separated the syntax and semantics of concept graphs. However, for the semantics she adopted the contextual semantics, i.e., the power context families, of Wille. Her syntax allows us to express existential quantification (with the so-called *generic marker*), and the description of situations and contexts (she allowed the *nestings of graphs*). Furthermore, she provided a sound and complete calculus. Therefore, the structure of her thesis can be outlined by the keywords syntax, semantics and calculus; hence it is closely related to the way mathematical logic is treated. But it has to be stressed that the *intention* of her thesis was different: The one-sided extensional view of mathematical logic and its focus on the truth values of formulas is refused, but her thesis is in line with contextual logic, i.e., a formalization of concepts, judgements and conclusions.

In Prediger's elaboration of concept graphs it is not possible to express negations, so this is the next step in the further development of concept graphs. Negations can occur on different levels: It should be possible to express the negation of concepts or relations, and it should be possible to express that judgements have the form of negations.

The negation of concepts is a difficult philosophical problem, so it is hard to implement a formal negation of concepts in FCA and concept graphs. In fact, there are at least two different possibilities for implementing negation on the formal concepts, namely on the side of their extents and on the side of their intents. The term 'negation' is used for a negation on the extensional side, for the intensional side the term 'opposition' is introduced (for this and the rest of this paragraph, see [77]). The next problem one has to face is that the negation or opposition of a formal concept is not a formal concept in general. So, in order to introduce negation on the conceptual level, the definition of formal concepts has to be generalized. This leads to the notion of so-called *semi-concepts* and *proto-concepts*.

In her thesis [29] (see also [28, 30]), Klinger has worked out the syntax and semantics for a negation on the conceptual level which is based on semi-concepts. Implementing negation on the basis of proto-concepts is investigated by Wille in [79] and [80].

In this treatise, how a negation on the level of judgements can be implemented will be elaborated. The main ideas that are needed can be found in the publications of Peirce (for existential graphs) and Sowa (for conceptual graphs). But a mathematical formalization is missing in these publications. Thus, in the tradition of the publications of Wille and Prediger, we provide a mathematization of conceptual graphs in this treatise, where an additional element of existential graphs, the so-called *cuts*[1], is added. Cuts allow us to express negation on the level of judgements. The resulting graphs will be called *concept graphs with cuts*. These graphs will allow us to express existential quantification too (thus, as the universal quantifier can be expressed with the existential quantifier and negations, universal quantification can be expressed as well), but we will not consider nested graphs. Similar to the thesis of Prediger, we will distinguish between syntax and semantics, and we will also provide a sound and complete calculus. Thus, in a sense, this treatise can be seen as a continuation of Prediger's thesis.

[1] The term *cut* is adopted from Peirce, where it denotes a syntactical device of existential graphs which is used to express negation. This term should not be mixed up with the term *cut* as used in mathematical logic.

In fact, as we can express relations between objects, conjunction and negation in judgements, and existential quantification, it will turn out that concept graphs with cuts are equivalent to first-order predicate logic. More precisely, it is the goal of this treatise to work out and mathematize a fragment of conceptual graphs which corresponds to first-order predicate logic. However, concept graphs with cuts are grounded on a contextual understanding of logic and are designed to fit in the framework of contextual logic. Hence, this treatise can show that '*Contextual Logic may reach at least the expressibility of first order predicate logic*' (see [77]).

July 2003

Frithjof Dau
Darmstadt, Germany

Contents

Start

Alpha

Beta

Appendix

1 Introduction

This treatise builds upon Sowa's theory of conceptual graphs ([59]), which in turn is based on of Peirce's system of existential graphs. For this reason we provide in the first two sections of this chapter short introductions into the these two graph systems.

Existential graphs have the power of first order logic, thus conceptual graphs claim to have at least the full power of first order logic, too. This assertion has already been stated by Sowa in [59], and in order to prove it, he provides mappings between conceptual graphs and first order logic and a calculus on the system of conceptual graphs. Unfortunately, a closer observation shows that his approach sometimes lacks preciseness, which lead to ambiguities, gaps and even mistakes. In the third section we present some of the flaws of Sowa's approach.

The concern of this treatise is to fix these flaws by a mathematization of conceptual graphs with negations. The resulting graphs are called *concept graphs with cuts*. The basic ideas of concept graphs with cuts will be presented the fourth section.

Finally, in Sect. 1.5, an overview for this treatise is provided.

1.1 Short Introduction to Existential Graphs

Conceptual graphs and in turn concept graphs are based on the existential graphs (EGs) of Charles Sanders Peirce (1839-1914). This is the reason why we try to provide a brief survey for EGs in this section. Of course this section can by no means replace a profound introduction into EGs. For those who want further information, we recommend [52], which is Roberts standard book on EGs, [56], a book where Shin provides her interpretation of EGs, [43], which is a tutorial of Peirce himself with commentary by Sowa, and [15], where the author offers a mapping from EGs into concept graphs.

Peirce invented the algebraic notation for predicate logic, namely the quantifiers. His notation is adopted from algebra, that is he used the symbols $\sum$ for existential quantification and $\prod$ for universal quantification, and

F. Dau: The Logic System of Concept Graphs with Negation, LNAI 2892, pp. 1-23, 2003.

he used products and sums to express conjunction and disjunction (see [52]). For example, Peirce used the formula $\sum_x \sum_y b_{xy} l_{xy} > 0$ to express '*that something is at once benefactor and lover of something*'. Another example is $\prod_x \prod_y (b_{xy} + l_{xy}) > 0$. Later on, Peano adopted this approach, but he introduced different symbols instead of Peirce's symbols. In particular, Peano introduced the symbols $\exists$ for existential quantification and $\forall$ for universal quantification, and $\wedge$ for conjunction and $\vee$ for disjunction. The Peano-notation is now the standard notation for mathematical logic. So the formula $\sum_x \sum_y b_{xy} l_{xy}$ of Peirce corresponds to the formula $\exists x.\exists y.(b(x,y) \wedge l(x,y))$ in the Peano-notation, and $\prod_x \prod_y (b_{xy} + l_{xy}) > 0$ corresponds to $\forall x.\forall y.(b(x,y) \vee l(x,y))$. Although Peirce invented the algebraic notation, he was not satisfied with this form of logic. As Roberts says in [52]: '*It is true that Peirce considered algebraic formulas to be diagrams of a sort; but it is also true that these formulas, unlike other diagrams, are not 'iconic' — that is, they do not resemble the objects or relationships they represent. Peirce took this for a defect.*' Peirce started his academic career in chemistry, and his knowledge of chemistry is the reason that he '*had become convinced that logic needed a more visually perspicuous notation, a notation that displayed the compound structure of propositions the way chemical diagrams display the compound structure of molecules.*' (see [22]). So Peirce developed different forms of diagrammatic logic, but is was not until 1896 that he discovered his existential graphs from which he said that they are '*the luckiest find of my career*' and which he called his '*Chef d'Œuvre'*. In fact, the system of existential graphs is a highly elegant system of logic which covers propositional logic, first order logic and even some aspects of higher-order logic and modal logic.

The system of EGs is divided into three parts: Alpha, Beta and Gamma. These parts presuppose and are built upon each other, i.e. Beta builds upon Alpha, and Gamma builds upon Alpha and Beta. Conceptual graphs are mainly based on Alpha and Beta, and only few aspects of Gamma are used, too. Hence we will provide only a short glance at Gamma.

We start with the description of Alpha. The EGs of Alpha consist only of predicate names of arity 0, which are called *medads*, and of closed, doublepoint-free curves which are called *cuts* and used to negate the enclosed subgraph. Medads can be considered as propositions. Propositions can be written down on an area (the term Peirce uses instead of 'writing' is *'scribing'*), and writing down a proposition is to assert it[1] The area where the proposition is scribed on is what Peirce called the *sheet of assertion*. It may be a sheet of paper, a blackboard or any other surface. Scribing several propositions next to each other (this operation is called a *juxtaposition*) asserts the truth of each proposition, i.e. the juxtaposition corresponds to

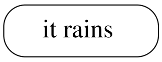
[1] An *assterted* proposition is called judgement.

the conjunction of the juxtaposed propositions. For example, scribing the propositions ‘it rains’ and ‘it is cold‘ next to each other yields the graph

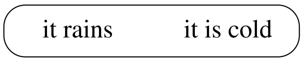
it rains   it is cold

which means ‘it rains and it is cold’.

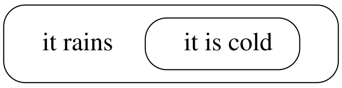
Encircling a proposition is to negate it. The line which is used to encircle a proposition is called a *cut*, the space within a cut is called its *close* or *area*. For example, the graph



has the meaning ‘it is not true that it rains’, i.e. ‘it does not rain’. The graph



has the meaning ‘it is not true that it rains and that it is cold’, i.e. ‘it does not rain or it is not cold’. Cuts may not overlap, but they may be nested. The next graph has two nested cuts.



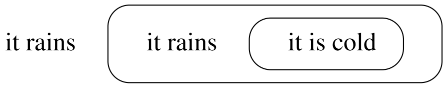
This graph has the meaning ‘it is not true that it rains and that it is not cold’, i.e. ‘if it rains, then it is cold’. The device of two nested cuts is called a *scroll*. From the last example we learn that a scroll can be read as an implication. A scroll with nothing on its first area is called *double cut* (it corresponds to a double negation). As mentioned before, the space within a cut is called the *area* of the cut. In the example above, we therefore have three distinct areas: All the space outside the outer cut, i.e. the sheet of assertion, the space between the outer and the inner cut, which is the area of the outer cut, and the space inside the inner cut, which is the area of the inner cut. An area is *oddly enclosed* if it is enclosed by an odd number of cuts, and it is *evenly enclosed* if it is enclosed by an even number of cuts.

As we have the possibility to express conjunction and negation of propositions, we see that Alpha has the expressiveness of propositional logic. Peirce also provided a calculus for existential graphs. For each system (Alpha, Beta, Gamma) this calculus has a set of five rules, which are named *erasure, insertion, iteration, deiteration, and double cut*, and only one axiom, namely the empty sheet of assertion. For Alpha, these rules are:

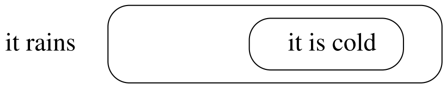
- Erasure: Any evenly enclosed graph may be erased.
- Insertion: Any graph may be scribed on any oddly enclosed area.

- Iteration: If a graph $\mathfrak{G}$ occurs on the sheet of assertion or in a cut, then a copy of the graph may be scribed on the same or any nested area which does not belong to $\mathfrak{G}$.
- Deiteration: Any graph whose occurence could be the result of iteration may be erased.
- Double Cut: Any double cut may be inserted around or removed from any graph of any area.

This set of rules is sound and complete. In the following, we provide a simple example of a proof (which will be an instantiation of modus ponens in EGs). Let us start with the following graph:



It has the meaning 'it rains, and if it rains, then it is cold'. Now we see that the inner instance of 'it rains' may be considered to be a copy of the outer instance of 'it rains'. Hence we can erase the inner instance of 'it rains' using the deiteration-rule. This yields:



This graph contains a double cut, which now may be removed. We get:

it rains it is cold

Finally we erase the proposition 'it rains' with the erasure-rule and get:

it is cold

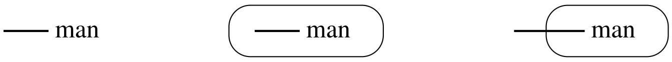
So the graph with the meaning 'it rains, and if it rains, then it is cold' implies the graph with the meaning 'it is cold'.

If we go from the part Alpha of EGs to the part Beta , predicate names of arbitrary arity may be used, and a new symbol, the *line of identity*, is introduced. Lines of identity are used to denote both the existence of objects and the identity between objects. Lines of identity are attached to predicate names. Peirce used to draw them bold. Consider the following graph:

man—— loves—— woman

It contains two lines of identity, hence it denotes two (not necessarily different) objects. The first line of identity is attached to the unary predicate 'man', hence the first object denotes a man. Analogously the second line of identity denotes a woman. Both lines are attached to the dyadic predicate 'loves', i.e. the first object (the man) stands in the relation 'loves' to the second object (the woman). The meaning of the graph is therefore 'there are a man and a woman such that the man loves the woman', or in short: A man loves a woman.

Lines of identity may cross cuts.[2] Consider the following graphs:



The meaning of the first graph is clear: it is 'there is a man'. The second graph is built from the first graph by drawing a cut around it, i.e. the first graph is denied. Hence the meaning of the second graph is 'it is not true that there is a man', i.e. 'there is no man'. In the third graph, the line of identity begins on the sheet of assertion. Hence the existence of the object is asserted and not denied. For this reason the meaning of the third graph is 'there is something which is not a man'.

Now we have the possibility to express existential quantification, predicates of arbitrary arities, conjunction and negation. Hence we see that the part Beta of existential graphs corresponds to first order predicate logic (that is first order logic with identity and predicate names, but without object names and without function names).

Essentially, the rules for Beta are extensions of the five rules for Alpha such that the Beta-rules encompass the properties of the lines of identity. For example, the erasure-rule and the insertion-rule are now formulated as follows:

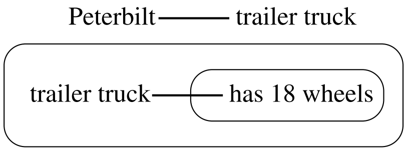
- Erasure: Any evenly enclosed graph and any evenly enclosed portion of a line of identity may be erased.
- Insertion: Any graph may be scribed on any oddly enclosed area, and two lines of identity (or portions of lines) oddly enclosed on the same area, may be joined.

For the formulation of the remaining three rules we refer to [52] (we have taken the other formulations of the rules from this source). But we illustrate the rules of Beta with an example which is taken from [62]. This example is a proof of the following syllogism of type Darii:

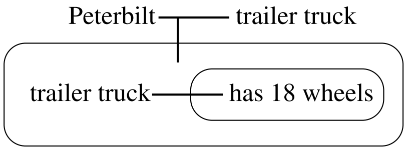
Every trailer truck has 18 wheels. Some Peterbilt is a trailer truck. Therefore, some Peterbilt has 18 wheels.

[2] This is not fully correct: Peirce denies that a LI may cross a cut (a corollary in in [42] states 'It follows that no line of identity can cross a cut.'). In Peirce's view, the third graph has *two* lines of identity which 'meet' at the cut. But the discussion of this needs a much deeper understanding of EGs and shall therefore be omitted here.

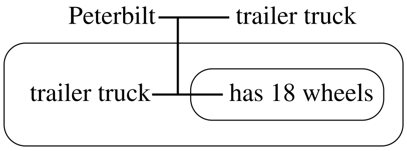
We start with an existential graph which encodes our premises:



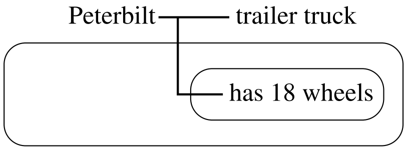
We use the rule of iteration to extend the outer line of identity into the cut:



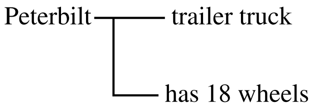
As the area of this cut is oddly enclosed, the insertion-rule allows us to join the loose end of the line of identity we have just iterated with the other line of identity:



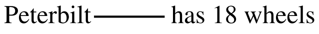
Now we can remove the inner instance of 'is a trailer truck' with the deiteration-rule:



Next we are allowed to remove the double cut:

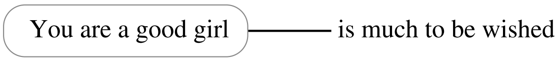


Finally we erase the remaining instance of 'is a trailer truck' and get:

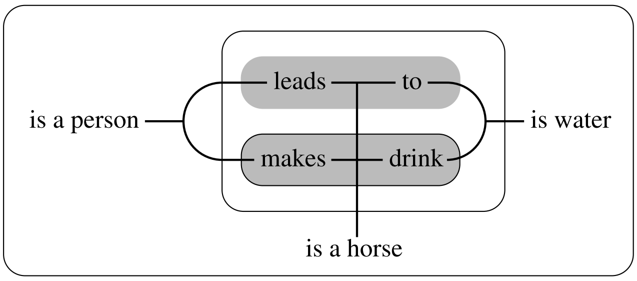


This is a graph with the meaning ‘some Peterbilt has 18 wheels’, hence with the conclusion of the syllogism.

The Gamma-part of EGs shall only be outlined here. Roughly spoken, it corresponds to higher order and modal logic, but as this system was never completed, it is difficult to be sure about Peirce’s intention. We will only explain one feature of Gamma which is essential for the development of conceptual graphs. The main idea of Gamma we want to describe is that a graph ‘*is applicable instead of merely applying it*’ (Peirce, quotation from [52]). In other words: Graphs, which have been used to speak *about* objects so far, can now in Gamma be treated like *objects themselves* such that other graphs speak about them.[3] A simple example for this idea can be found in [41], where Peirce says: ‘*When we wish to assert something about a proposition without asserting the proposition itself, we will enclose it in a lightly drawn oval, which is supposed to fence it off from the field of assertions.*’ Peirce provides the following graph to illustrate his explanation:



The meaning of this graph is: ‘You are a good girl’ is much to be wished. Peirce generalized the notion of a cut. The lightly drawn oval is not used to negate the enclosed graph, it is merely used to ‘*fence it off from the field of assertions*’ and to provide a graphical possibility to speak about it. Peirce extended this approach further. He started to use colours or tinctures to distinguish different kind of context. For example, he used the colour red to indicate possibility (which is – due to the fact that this treatise is printed in black and white – replaced by gray in the next example).



In this example we have two red (gray) ovals. One is purely red; it says that the enclosed graph is possible. The other one is a cut which is red inside, hence it says that the enclosed graph is *im*possible. As the three lines of identity start in the area of a scroll, they can be understood as *universally* quantified objects. Hence the meaning of the graph is: For all persons, horses

[3] This is a kind of abstraction which Peirce called ‘hypostatic abstraction.’

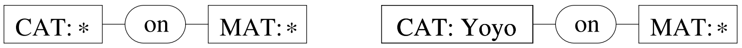
and water, it is possible for the person to lead the horse to the water, but is impossible to make the horse drink. Or, for short: You can lead a horse to water, but you can't make him drink.

It is important to note that Peirce did not consider the tinctures to be logical operators, but to be meta-level operators. That is, they are part of a metalanguage which can be used to describe how logic applies to the universe of discourse. Peirce said himself: '*The nature of the universe or universes of discourse (for several may be referred to in a single assertion) in the rather unusual cases in which such precision is required, is denoted either by using modifications of the heraldic tinctures, marked in something like the usual manner in pale ink upon the surface, or by scribing the graphs in coloured inks.*' (quotation taken from [66]).

1.2 Short Introduction to Conceptual Graphs

Sowa invented conceptual graphs on the basis of existential graphs. In [62] he says '*Conceptual graphs (CGs) are an extension of existential graphs with features adopted from linguistics and AI. The purpose of the system is to express meaning in a form that is logically precise, humanly readable, and computationally tractable.*' The term 'extension' is to be understood semantically. But it is not to be understood syntactically: Sowa adopted the ideas of existential graphs, but conceptual graphs have a different and much richer syntax. As Sowa explains this in [62]; '*Besides Peirce's primitives, conceptual graphs provide means of representing case relations, generalized quantifiers, indexicals and other aspects of natural languages.*' Two other fundamental aspects (which are perhaps so fundamental that Sowa did not even mention them explicitely in the quotation above) are the following: First, in conceptual graphs different kinds of referents exist, in particular names for objects. Second, conceptual graphs may serve as referents themselves. This is called *nesting* of conceptual graphs (corresponding to the nesting of cuts in existential graphs). Both aspects will be explained in the rest of this section.

In order to illustrate the broad range of conceptual graphs, we will provide several examples. The first two are:



The meanings of this graph are 'a cat is on a mat' and 'the cat Yoyo is on a mat', respectively. In each concept box, we have a *type label t* and a *referent r*. Sowa calls the boxes *concepts*, but in this treatise, the term *concept box* will be used instead. Furthermore, when we later introduce concept graphs, we will have *concept names* instead of type labels. In the conceptual graph above, we have the types CAT and MAT. The referents are 'Yoyo' and the *generic marker* '∗'. Type labels are ordered in a sub-type-relation, according to their level of generality. For example, CAT is a subtype of ANIMAL. An ordered set of types is often called *support*, *type hierarchy*, *taxonomy* or even *ontology*. Type hierarchies usually contain a greatest type ⊤, the *universal type*, which contains every object (of the respective universe of discourse) in its extension. In the graph above, the two concept boxes contain two different kinds of referents. The referent 'Yoyo' of the first box is a name for an object. The referent '∗' of the second box is a fixed symbol which does not denote a particular individual, but it denotes an individual which is not further specified. So the star '∗' can be read as an existential quantifier. It is called *generic marker*, hence the second box is called *generic concept box*. The ovals are *(conceptual) relations* between the referents of the concept boxes which are linked to the oval.

Sowa uses various kinds of referents, for example

– names: $\boxed{\text{PERSON: John}}$, $\boxed{\text{DOG: Lucky, Matula}}$,

– quantifiers: $\boxed{\text{MEAT: } *}$ (some meat), $\boxed{\text{DOG: } \forall}$ (all dogs),

– measure specifications: $\boxed{\text{TIME-PERIOD: @5 seconds}}$, $\boxed{\text{Money:@\$5}}$ or $\boxed{\text{WATER: @5 liters}}$,

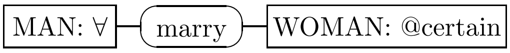
– control marks ?, !, #: $\boxed{\text{DOG: ?}}$ (which dog?), $\boxed{\text{Dog: Lucky !}}$ (emphasis on Lucky), $\boxed{\text{MEATBALL: \#}}$ (*the* meatball),

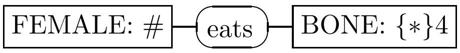
– generic plurals: $\boxed{\text{BONE: } \{*\}}$ (some bones) or $\boxed{\text{BONE: } \{*\}4}$ (four bones),

– prefixes: $\boxed{\text{PERSON: Dist\{Bill, Mary\}@5}}$ (five persons distributively including Bill, Mary, and others), or $\boxed{\text{LADY: Cum}\{*\}}$ (a set of Ladies).

(all examples taken from [60]).

So the following graphs have the meanings 'all men married a certain woman' and 'she is eating four bones'.





It is clear that these various kinds of referents go beyond the expressiveness of first order logic. In particular they are not adopted from EGs. But even in the first example of this section (i.e., the graph with the meaning 'a cat is on a mat') we can already see a crucial difference between existential graphs and conceptual graphs. The lines of identity in existential graphs serve different purposes: They are used for existential quantification, they are used to connect arguments to relations, and they are used to show identity between arguments. In conceptual graphs, all these functions are separated. Existential quantification is expressed by generic concept boxes. Referents of concept boxes serve as arguments for relations, and the connection between the arguments and the relations is drawn by relation ovals. Now we have to clarify how identity is expressed in conceptual graphs. For this, a new syntactical element is used. In [60], Sowa says '*Two concepts that refer to the same individual are coreferent. [...] To show that they are coreferent, they are connected with a dotted line, called a* coreference link.' He illustrates this using the graph of Fig. 1.1.

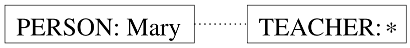


Fig. 1.1. conceptual graph for 'Mary is a teacher'

It says that Mary is a person and there is a teacher who is the same as Mary, or for short: Mary is a teacher.

The second fundamental aspect of conceptual graphs which shall be explained here is the nesting of conceptual graphs. When graphs are nested, some conceptual graphs serve as referents in concept boxes of other conceptual graphs themselves. In this manner, some conceptual graphs make statements and assertions about other conceptual graphs, i.e., the system of conceptual graphs offers the possibility of meta-level statements. Concept boxes whose referents are conceptual graphs are called *contexts*.[4] Peirce's sheet of assertion or the surface where a conceptual graph is scribed on can as well be understood as a context, the *outermost context*.

In Fig. 1.2 we present a well-known example of a nested conceptual graph: It contains two contexts, namely the concept boxes of type PROPOSITION and SITUATION (which are common types of contexts). The graph can be read as follows: The person Tom believes a proposition, which is described by a graph itself. The proposition says that the person Mary wants a situation, which again is described by a graph. In this situation we have a concept box $\boxed{\top : *}$ which is connected with a coreference link to the concept box

[4] The term 'context' occurs in several meanings and implementations in logics, linguistics or artificial intelligence. But, as Sowa says in [66]: '*The notion of context is indispensable for any theory of meaning, but no consensus has been reached about the formal treatment of context.*' An introduction into the various ideas of contexts is beyond the scope of this treatise (we refer to [66] or Chap. 5 of [65] for an introduction and discussion of contexts in conceptual graphs and recommend the treatises of McCarthy, Barwise and Perry for further reading).

$\boxed{\text{PERSON:Mary}}$ in the context above. So the situation is that Mary marries a sailor. The formal understanding of the whole graph is now: The person Tom believes the proposition that the person Mary wants the situation that Mary marries a sailor. In short: Tom believes that Mary wants the situation in which she marries a sailor, or even shorter: Tom believes that Mary wants to marry a sailor.

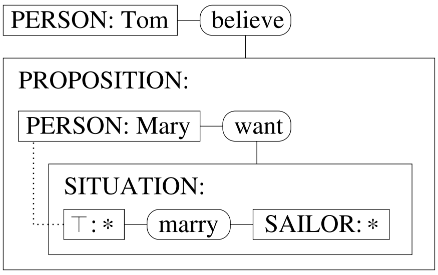


Fig. 1.2. a nested conceptual graph

In the last section, we said that Peirce used coloured or tinctured cuts to distinguish different kinds of contexts. So when an existential graph appears in a specific context, this context is drawn as an oval or coloured area. This oval is used '*to fence it (the existential graph) off from the field of assertions*', and the colour of the cut specifies the type of the context. Sowa adopted this idea of a graphical representation of contexts, and we see that the tinctured cuts of existential graphs correspond to the contexts (i.e. concepts with complete graphs as referents) in conceptual graphs.

A crucial aspect for the motivation of this treatise (we will discuss this in the next section and in Sect. 14.1) is that Sowa expresses negations with contexts, too: '*The EG negative contexts are a special case of the CG contexts. They are represented by a context of type Negation whose referent field contains a conceptual graph that states the proposition which is negated.*' (taken from the definitions in [62] which are based on the draft ANSI standard). Concept boxes of type NEGATION are introduced by Sowa as abbreviations for contexts of type proposition with an unary relation 'NEG' attached to it (see [60], where he says that '*Negation (NEG) is one of the most common relations attached to contexts*'), and Sowa often abbreviates these contexts by drawing a simple rectangle with the mathematical negation symbol ¬ (e.g. in [64]).

A well known example for a conceptual graph with negations is presented in Fig. 1.3. The device of two nested negation contexts corresponds to the scrolls in existential graphs and can be understood as an implication. Hence, the meaning of the graph is ‘if a farmer owns a donkey, then he beats it’.[5]

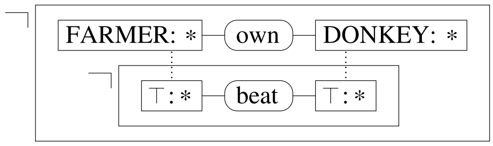


Fig. 1.3. a conceptual graph with negations

Sowa says that conceptual graphs are an extension of existential graphs. How the term ‘extension’ is to be understood is explained in [59]: *‘existential graphs [...] form the logical basis for conceptual graphs:*

- *They have the full power of first order logic.*
- *They can represent modal and higher-order logic.*
- *The rules of inference are simple and elegant.*
- *The notation is easily adapted to conceptual graphs.’*

In particular conceptual graphs are designed to have at least the full power of first order logic, that is first order logic with identity, object names and predicate names, but without function names. In the following, we will use the abbreviation FOL for this style of logic. In [59] we find: ‘*Any formula in first-order logic can be expressed with simply nested contexts and lines of identity.*’ But the system of conceptual graphs is richer than first order logic: In [67] Sowa says that ‘*CGs have been developed as a graphic representation for logic with the full expressive power of first-order logic and with extensions to support metalanguage, modules, and namespaces.*’

To show how conceptual graphs and first order logic are related, Sowa defines a formal operator Φ which maps conceptual graphs to first order logic. The definition of Φ can be found in [59], and we find various comments and examples among the treatises of Sowa. In [60] Sowa describes the value of Φ as follows: ‘*Since conceptual graphs form a complete system of logic with their own rules of inference and model-theoretic semantics, there is no need to map them to any other version of logic. Yet the definition of the Φ operator that maps conceptual graphs to predicate calculus is interesting for several reasons: it explicitly shows the structural differences between the two systems, it enables theoretical advances made with one system to be adapted to the other,*

[5] This examples well known and used for other purposes as well. Particulary, the problems in natural language processing caused by anaphora are discussed in Kamps Discourse Representation Theory on this example, but these problems are not treated in this treatise.

and it allows programs that use one system to interact with programs based on the other. The mapping Φ, however, is only defined for features that can be represented in predicate calculus.' For example, he translates the graph of Fig. 1.1, p. 10, with the meaning 'Mary is a teacher' to

$$\exists x.(PERSON(Mary) \land TEACHER(x) \land Mary = x) \quad ,$$

and the graph Fig. 1.3 with the meaning 'if a farmer owns a donkey, then he beats it' is translated to

$$\neg\exists x.\exists y.(FARMER(x) \land DONKEY(y) \land owns(x,y) \land \neg beat(x,y)) \quad .$$

The last formula can be logically transformed using to the simplified formula $\forall x.\forall y.((FARMER(x) \land DONKEY(y) \land owns(x,y)) \to beat(x,y))$.

When he developed the system of conceptual graphs, Sowa did not only adopt the iconic idea and many syntactical and semantical elements of existential graphs. He also adopted the calculus of existential graphs . In [59] he provides a calculus which is a one-to-one translation of Peirce's five rules into the system of conceptual graphs. In [62] or in [67], this system is refined, and he presents a set of six rules, grouped into three sets of two rules each. The following description of these rules is taken from [62]:

1. Equivalence rules*: copy and simplify. The copy rule copies a graph or subgraph. The simplify rule performs the inverse operation of erasing a copy.*
2. Specialization rules*: restrict and join. The restrict rule specializes the type or referent of a single concept node. The join rule merges two concept nodes to a single node.*
3. Generalization rules*: unrestrict and detach. The unrestrict rule, which is the inverse of restrict, generalizes the type or referent of a concept node. The detach rule, which is the inverse of join, splits a graph in two parts at some concept node.*

Using these six rules, Sowa provides a calculus for conceptual graphs which is still based on Peirce's calculus. The following is again an excerpt of [62]:

By handling the syntactic details of conceptual graphs, the generalization and specialization rules enable Peirce's rules of inference to be stated in a general form that is independent of the graph notation.[...]

1. Erasure: *In a positive context, any graph u may be replaced by a generalization of u; in particular, u may be erased (i.e. replaced by the blank, which is a generalization of every CG).*
2. Insertion: *In a negative context, any graph u may be replaced by a specialization of u; in particular, any graph may be inserted (i.e. it may replace the blank).*

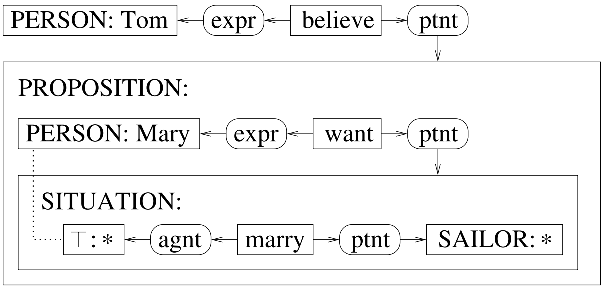
3. Iteration: *If a graph u occurs in a context c, another copy of u may be drawn in the same context c or in any context nested in c.*
4. Deiteration: *Any graph u that could have been derived by iteration may be erased.*
5. Equivalence: *Any equivalence rule (copy, simplify, or double negation) may be performed on any graph or subgraph in any context.*

1.3 Problems with Conceptual Graphs

In some sense the system of conceptual graphs is not fixed, but open-minded. It is designed to be used in fields like software specification and modelling, knowledge representation, natural language generation and information extraction, and these fields have to cope with problems of implementational, mathematical, linguistic or even philosophical nature. To cope with such problems, different modifications and extensions of conceptual graphs are suggested. For this reason it is very hard and perhaps not even desirable to provide a definition which covers all possible aspects and features of conceptual graphs. On the other hand this leads to several difficulties and fallacies, ranging from lacks of preciseness and ambiguities over minor gaps to major mistakes and contradictions, in and between different notations or implementations of conceptual graphs. This has already been recognized by some authors, for example by Chein and Mugnier (see [7]), and Wermelinger who has objections against Sowa's definition of Φ and his calculus. He says in [70]) '*that Sowa's original definition of the mapping (Φ) is incomplete, incorrect, inconsistent, and unintuitive, and the proof system is incomplete, too.*' Although this judgement is formulated very harshly, it must be acknowledged from a mathematical point of view. In this section we will show some of the problems appearing in the system of conceptual graphs.

Sowa wants to provide at least for a core of conceptual graphs a precise definition, named *abstract syntax* (see [64], [67]). This definition was intended to result in an ISO-standard, but it is still in the making. In the working draft [67] for the definition he writes: '*Informally, a CG is a structure of concepts and conceptual relations where every arc links a concept node and a conceptual relation node. Formally, the abstract syntax specifies conceptual graphs as mathematical structures without making any commitments to any concrete notation or implementation.*' But unfortunately, the definition of the abstract syntax is not a mathematical definition, but it is written in common English. Furthermore, the definitions are incomplete, and many aspects of conceptual graphs which should be covered by the definition are only explained in the comments (for example, Sowa assigns to each relation node a *signature*, but he explains only in the comments that these signatures serve to represent constraints on the types of the attached concept boxes. But these constraints are not part of the definition.) In particular the definition of Sowa does not suffice to derive a mathematical definition from it.

The same holds for the definition of the Φ-operator. Φ is not defined on the *whole* set of conceptual graphs. As Sowa says, it ‘*is only defined for features that can be represented in predicate calculus*’, but he does not specify what features he talks about. In fact, he applies this operator to graphs with different kinds of quantifiers or with nestings, when it is not even clear if these graphs can be translated to FOL. In [60], he provides the following graph for the proposition ‘Tom believes that Mary wants to marry a sailor‘ (this graph is more complicated than the first graph we provided for the same proposition, because it explicates the linguistic roles of the verbs):



Afterwards, he translates this graph to the following formula:

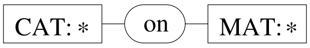
$$\begin{aligned}&\exists x.Person(Tom) \wedge believe(x) \wedge expr(x, Tom) \wedge ptnt(x,\\&\quad \exists y.\exists z.PERSON(Mary) \wedge want(y) \wedge SITUATION(z) \wedge\\&\quad expr(y, Mary) \wedge ptnt(y, z) \wedge descr(z,\\&\qquad \exists u.\exists v.(marry(u) \wedge SAILOR(v) \wedge agnt(u, Mary) \wedge ptnt(u, v))))))\end{aligned}$$

In this formula, whole formulas occur as arguments of relations, namely of *ptnt* (patient) and *descr* (description), which Sowa uses to express nestings in FOL. But using formulas as arguments in relations is a feature beyond FOL, so this translation cannot be considered as translation into FOL.

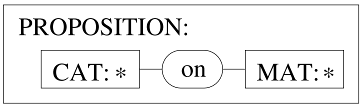
In [70], Wermelinger reveals some further mistakes in the definition of Φ.

The handling of negation yields another class of problems. In his Cambridge Lectures ([41]), where Peirce provided the existential graph with a meta-level statement for the sentence *‘you are a good girl’ is much to be wished* (see p. 7), he also said: ‘*That a proposition is false is a* logical *statement about it, and therefore in a logical system deserves special treatment.*’ In conceptual graphs, negations are implemented as special contexts, and contexts are used to draw meta-level propositions on conceptual graphs. Hence, the character of negation as logical operator in existential graphs changed to a meta-level character in conceptual graphs. In fact, Sowa says in [66]: ‘*To support first-order logic, the only necessary meta-level relation is* negation.’ But expressing negation as a meta-level operator leads to difficulties. Negation is a special operation, and its properties have to be captured by the semantics and by any calculus. To understand the difficulties that can appear, we consider the following two conceptual graphs:

$\mathfrak{G}_1 :=$

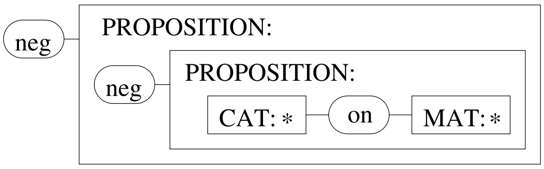


, $\mathfrak{G}_2 :=$



The meaning of the first graph is well known: It is 'a cat is on a mat'. The meaning of the second graph is, strictly speaking, 'there exists a proposition, which states that a cat is on a mat'. Quoting a proposition changes its character: there is a crucial difference between asserting and quoting a proposition, between using and mentioning a linguistic item. In particular these two graphs have different meanings. Now we consider the graph below:

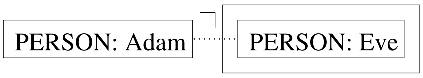
$\mathfrak{G}_3 :=$



$\mathfrak{G}_3$ contains a double negation, hence the double cut rule yields that $\mathfrak{G}_1$ and $\mathfrak{G}_3$ are equivalent. On the other hand, $\mathfrak{G}_3$ contains a proposition which quotes a proposition which quotes a graph, i.e. we have a meta-meta-statement. For this reason it seems odd that $\mathfrak{G}_1$ and $\mathfrak{G}_3$ have the same meaning.

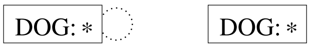
A more detailed analysis of expressing negation in conceptual graphs is given in Sect. 14.1 of the appendix.

Expressing identity in conceptual graphs with coreference-links leads to another class of difficulties. In particular, the meaning of coreference-links connected to a concept box within a negation is not straightforward. This shall be illustrated by the following example:

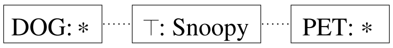


The link in the graph which expresses the identity between Adam and Eve looks symmetric. But it is crucial whether the equating of Adam and Eve takes place *inside* or *outside* the negation context. In the first case, the graph translates to $PERSON(Adam) \wedge \neg(PERSON(Eve \wedge (Adam = Eve))$, hence to a valid formula (of course we assume that Adam and Eve are different persons). In the second case, the graph is translated to the formula $PERSON(Adam) \wedge (Adam = Eve) \wedge \neg PERSON(Eve)$, which is a non-valid formula. It is agreed that equality is always placed in the inner context, hence the graph translates to the first, valid formula. But this is not clear from the graphical representation of coreference links.

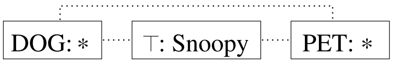
The next point is that the properties of identity, i.e. the properties of coreference-links, have to be captured by any calculus in a sound and complete way. For example, if the following graph on the left is a well-formed conceptual graph (which is not clear from the definition of conceptual graphs), it should be provably equivalent to the conceptual graph on the right:



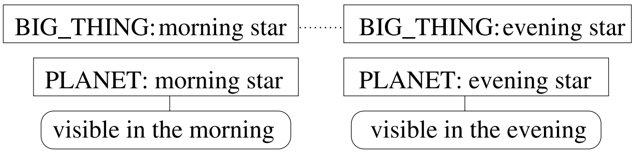
As identity is a transitive relation, the next two graphs should be provably equivalent, too:



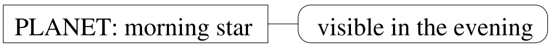
and



Finally, given the graph



it should be possible to derive



Sowa did not prove that his calculus allows derivations like this, and it is difficult to decide from the definition of his calculus whether these derivations are possible.

As for negation, a more detailed analysis of expressing identity in conceptual graphs is given in Sect. 14.2 of the appendix.

Finally, the calculus has to cover the special meaning of the universal type $\top$. For example, it should be allowed to derive $\boxed{\top : \mathrm{Tom}}$, if Tom is a valid object name, and it should be possible to derive $\boxed{\top : *}$, too. This seems to be impossible applying the calculus of Sowa.

To summarize: There are objections against the definition of conceptual graphs, against the definition of the Φ-operator, and against the implementations of negation and identity. The main reasons for these objections are:

- Lack of preciseness and informal definitions, which yield gaps and mistakes in the syntax, and in the calculus.

- Gaps in the implementation of the special concept name ⊤ and in the implementation of identity.
- Implementing negation as a meta-level operator, which yields difficulties in the semantics of conceptual graphs.

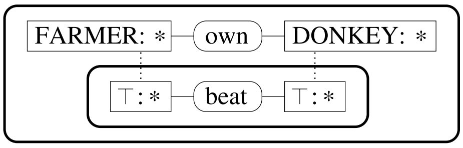
1.4 Short Introduction to Concept Graphs with Cuts

In the last section we have seen that conceptual graphs are designed to have the full expressive power of first-order logic. But there are several gaps and mistakes in syntax, semantics and in the calculus of conceptual graphs and in the definition of the operator Φ. Furthermore, Sowa fails to explain how it can be *proven* that conceptual graphs have the expressive power of first-order logic.

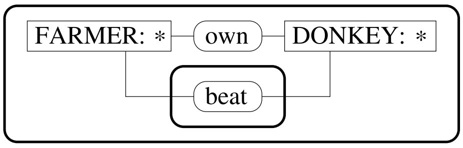
The concern of this treatise is to amend these flaws. To be more precise:

> It shall be mathematically elaborated a fragment of conceptual graphs which is equivalent to first order logic, i.e., all necessary definitions shall be provided, and the equivalence shall be formulated and proven. In this treatise, we will introduce so-called 'concept graphs with cuts' and will show that they are exactly such a required fragment.

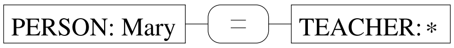
To express formulas of FOL with conceptual graphs, we have to use negation contexts. Other contexts are not needed. Since we have argued that treating negation as a meta-level operator results in problems, we will consider negation as a logical, not a meta-level operator. As Peirce said, negation deserves special treatment. The first idea is: Instead of using negation contexts of conceptual graphs, a new syntactical element is needed to express negation. One cannot help having the idea to use the cuts of existential graphs for this purpose. To distinguish these ovals from the ovals which are drawn around relation names, we propose drawing them bold. So, a graph with the meaning 'if a farmer owns a donkey, then he beats it' which uses cuts instead of negation contexts would look like this:



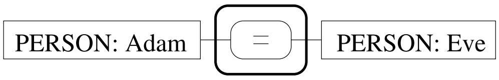
In contrast to the use of contexts, there is no reason to draw cuts only around 'complete' subgraphs. We can improve the usability of cuts if we allow the cuts to be drawn around arbitrary parts of a conceptual graph. Then the graphical representation of cuts is more easily readable and intuitively understandable. For example, instead of the graph above, we can draw the following graph with the same meaning:



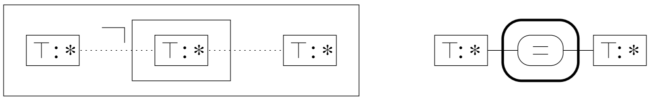
Another objection we had is the use of coreference links to express identity. We suggest to implement identity, like any other relation, with relation ovals which are labelled with the name '$\doteq$'. These relation ovals are therefore called *identity links*. For example, the following graph expresses that Mary is a teacher:



The advantages of this approach can be seen best if cuts and identity links are combined. Using identity links and cuts which may be drawn around single relation ovals, we can express the sentence 'The persons Adam and Eve are different' with the following graph:



The graphs which express 'there are at least two things' are another example. The left graph is the example of Sowa which uses coreference links and a negation context, the right one is a graph which uses a cut and an identity link.



These examples show that using cuts and identity links yield graphs which can be read more easily than their counterparts which use negation contexts and coreference links.

After replacing negation contexts by cuts and coreference links by identity links, the next step is the *mathematization* of these new graphs.

The first attempt of a mathematization of conceptual graphs has been made by Chein and Mugnier in [7]. In [75] Wille adopted and slightly modified this approach and combined the systems of Formal Concept Analysis and conceptual graphs. To distinguish between the conceptual graphs of Sowa and his mathematization of the graphs, he coined the term *concept graph*, i.e., concept graphs are the mathematization of conceptual graphs. This approach has been extended in several publications, mainly of Wille (e.g. [76]) and Prediger (see [44] and [45]), and this treatise builds upon this approach, too. As we will provide a mathematization of conceptual graphs where we have added the cuts of existential graphs, we will call these graphs *concept graphs with cuts*. Now, in order to provide a mathematical foundation of concept graphs with cuts and an elaboration of the equivalence between the system of concept graphs with cuts and FOL, the following steps must be taken:

1. Provide a mathematical definition of concept graphs with cuts and identity links.
2. Provide a mathematical definition of a (contextual) semantics for concept graphs with cuts.
3. Provide a mathematical definition for the operator Φ which maps concept graphs with cuts to FOL-formulas, and provide a mathematical definition for an operator Ψ in the inverse direction, i.e., Ψ maps FOL-formulas to concept graphs with cuts.
4. Provide a mathematical definition for a calculus on the system of concept graphs with cuts.
5. Provide a proof that the calculus is sound and complete.
6. Provide a proof that the logical systems of concept graphs with cuts and FOL with its calculus are equivalent via the operators Φ and Ψ.

The steps mentioned above are the major tasks which this treatise means to perform.

1.5 Outline of This Treatise

This treatise is divided into four main parts: Start, Alpha, Beta, and the Appendix. In the first part, a motivation and an overview of this treatise is presented. Furthermore, in Chap. 2, the basic definitions for the syntax of concept graph are provided.

The crucial parts of this treatise are Alpha and Beta. The partition into Alpha and Beta is adopted from the system of EGs. In the part Alpha, only concept graphs with cuts which do not contain generic markers (so-called *nonexistential* concept graphs with cuts) are investigated. In Beta, the approach of Alpha is extended to concept graphs with cuts *with* generic markers.

We have to admit that the comparison of nonexistential concept graphs with cuts with the part Alpha of EGs is not fully correct. We have seen that the part Alpha of EGs corresponds to propositional logic, but nonexistential concept graphs with cuts correspond to FOL-formulas without variables. As the relations between objects can be expressed in these formulas, this part of logic has a higher expressive power than propositional logic. But as we cannot *range* over objects, the expressive power of this logic is strictly weaker than FOL. So the power of nonexistential concept graphs is placed *between* propositional and first order logic, hence between the part Alpha and the part Beta of EGs. However, in order to distinguish between the parts of this treatise for nonexistential and for existential concept graphs with cuts, we adopted the terms 'Alpha' and 'Beta'.

The proceeding in Alpha is straight forward and can be outlined with the terms 'syntax, semantics, calculus'. As the syntax for Alpha is already defined in Chap. 2, first a contextual semantics, which is based on power context families, is presented. Afterwards, a calculus which is based on the Alpha-calculus of Peirce is provided. It will be shown that the calculus is sound and complete.

This calculus contains – besides the concept graph versions of Peirce's rules – some additional rules. In Sect. 1.3 (see pp. 15 to 17) we have argued that it is not sufficient to translate Peirce's rules to the system of conceptual graphs: The resulting calculus will generally not be complete. This can now be proven: In Sect. 6.3 we show that the calculus is strictly weakened if one of the additional rules is removed.

The part Beta is more complex than the part Alpha. We add generic markers to concept graphs with cuts, and for this reason we have to extend the definition of the semantics as well as the calculus. What is new in Beta is that we provide the definitions of the mappings Φ, which maps concept graphs with cuts to FOL-formulas, and of Ψ in the inverse direction, i.e., Ψ maps FOL-formulas to concept graphs with cuts. In the part Beta it will be shown that the semantics, the calculus, and the mappings fit perfectly together. In particular we will show that Φ and Ψ are mutually inverse isomorphisms (up to equivalence of concept graphs with cuts resp. formulas) between the logical systems of concept graphs and FOL. And we will show that the Beta-calculus is sound and complete, too. In contrast to Alpha, we will not prove the completeness directly, but we will apply the appropriate result for FOL.

More detailed overviews for Alpha and Beta can be found in Chap. 3 for Alpha and Chap. 7 for Beta.

It has already been stated that the goal of this treatise is to elaborate the fragment of conceptual graphs which is equivalent to first order logic. For this reason we have to emphasize that the *theorems* of this treatise are by no means surprising, on the contrary they had to be expected. This treatise is a mathematization of the mentioned fragment. Hence the true value of this treatise is not to be found in the *results*, but in the *definitions*. The results which will be proven simply show that the mathematization is 'correct'.

We have explained why the negation contexts of conceptual graphs should be replaced by the cuts of EGs, and why coreference links should be replaced by identity links. There is no absolute *need* for these replacements: They have been *decisions* in the process of mathematization. Of course, these decisions are be open to discussion, and we have to give the reasons for them. In the Appendix, in Chap. 14, we provide reasons for the use of cuts and identity links, for the requirement to consider only concept graphs with dominating nodes, and for the design of the calculus.

1.6 Acknowledgements

This treatise is my PhD-thesis and as nearly all (at least I guess so) PhD-theses it could only come into being with the ideas and help of many other people.

First of all, I want to express my deep gratitude to Rudolf Wille, the supervisor of this thesis, for his generous encouragement, inspiration and support. His visions and insight broadened my horizon considerably.

I would like to thank very much Gerd Stumme, Susanne Prediger and Julia Klinger for carefully proofreading my thesis and for their helpful suggestions.

To the aforementioned and Peter Burmeister, Joachim Hereth Correia, Björn Vormborck, Richard Holzer and Markus Helmerich – the other members of my working group – of the department of mathematics – I owe a dept of gratitude for many interesting discussions and all their suggestions every time I presented new results of my work in our seminar. There was not one session that did not give impetus to further work. I admired Peter Burmeister for his patience. He simply refused to let himself get worked up, even if when I got impatient, and by doing so made me devote attention to details and helped me to close gaps.

Michael Richter, my secondary supervisor, helped me see my work in a larger context by giving valuable information as to how my results fit in with the results in related fields. For his assistance and the careful correction of my thesis, I want to thank him cordially.

I would like to thank Thomas Streicher: I enjoyed and greatly profited from the often controversial, but always fair discussions with him.

Reading Sec. 1.3 could lead to the impression that I think little of John Sowa's work. Therefore I want to stress – hence and now – that quite the reverse is the case. His visions opened a field of research of his own: the field of conceptual graphs. His ideas and work are a major source of inspiration for my thesis.

Rudolf Wille as well as John Sowa are greatly influenced by Charles Sanders Peirce' philosophy, and it is his work that my thesis is based upon above all. Mainly his existential graphs inspire my deep respect. It is simply adorable that he developed such a well thought out and elegant system of logic more than 100 years ago.

I would like to thank my parents who made it possible for me to study mathematics. Without them I would not be, where I am now.

My very special thank to Reinhild Trompke, the most important person in my life, for supporting me both emotionally and mentally and for covering me by taking care of the petty problems of every day life while I had my head in the clouds.

You are surprised that in a few passages the English is better than in the others? It is thanks to Ingrid Merz who translated these paragraphs, as I was under a lot of pressure at the time.

Last not least I am quite grateful to the Springer-Verlag. It makes me feel proud and a little awkward that this renowned press offered to publish my work.

2 Basic Definitions

Concept graphs as mathematically defined structures were first presented by Wille in [75]. Prediger separated the syntax and semantics of concept graphs. In [45], she defined *directed multi-hypergraphs* which are bipartite graphs with vertices and edges. These graphs are the underlying structure of concept graphs: From these graphs, concept graphs are obtained by additionally labelling the vertices and edges with names for objects, concepts and relations. Later on, Wille replaced the term 'directed multi-hypergraph' was replaced by the simpler term 'relational graph' (this term is already used in [28] and [29]). We adopt this notation and extend it to include

- the possibility to express negations by adding the cuts of EGs, and
- the possibility to express identity by using a special binary relation $\doteq$.

These graphs will be called *relational graph with cuts* . Their definition, especially the notions of *cuts* and *areas*, is closely related to the ideas of Peirce. Analogously to Prediger, we label the vertices and edges of concept graphs with cuts with object names, concept names and relation names. The resulting graphs are called *(simple) concept graphs with cuts.*

2.1 Relational Graphs with Cuts

In this section, the basic definitions and properties for relational graphs with cuts are presented.

Definition 2.1 (Relational Graphs with Cuts).

A structure $(V, E, \nu, \top, Cut, area)$ *is called a* relational graph with cuts *if*

- *V, E and Cut are pairwise disjoint, finite sets whose elements are called* vertices*,* edges *and* cuts*, respectively,*
- $\nu : E \to \bigcup_{k\in\mathbb{N}} V^k$ *is a mapping*[1]*,*

[1] We set $\mathbb{N} := \{1, 2, 3, \ldots\}$ and $\mathbb{N}_0 := \mathbb{N} \cup \{0\}$.

F. Dau: The Logic System of Concept Graphs with Negation, LNAI 2892, pp. 25-38, 2003.

- $\top$ *is a single element with* $\top \notin V \cup E \cup Cut$*, called the* sheet of assertion*, and*
- $area : Cut \cup \{\top\} \to \mathfrak{P}(V \cup E \cup Cut)$ *is a mapping such that*

  a)$c_1 \neq c_2 \Rightarrow area(c_1) \cap area(c_2) = \emptyset$,

  b)$V \cup E \cup Cut = \bigcup_{d \in Cut \cup \{\top\}} area(d)$,

  c)$c \notin area^n(c)$ *for each* $c \in Cut \cup \{\top\}$ *and* $n \in \mathbb{N}$ *(with* $area^0(c) := \{c\}$ *and* $area^{n+1}(c) := \bigcup\{area(d) \mid d \in area^n(c)\}$*).*

For an edge $e \in E$ *with* $\nu(e) = (v_1, \ldots, v_k)$ *we set* $|e| := k$ *and* $\nu(e)\big|_i := v_i$*. Sometimes, we will write* $e\big|_i$ *instead of* $\nu(e)\big|_i$*, and* $e = (v_1, \ldots, v_k)$ *instead of* $\nu(e) = (v_1, \ldots, v_k)$*. We set* $E^{(k)} := \{e \in E \mid |e| = k\}$*.*

For $v \in V$ *let* $E_v := \{e \in E \mid \exists i.\nu(e)\big|_i = v\}$*. Analogously, for* $e \in E$ *let* $V_e := \{v \in V \mid \exists i.\nu(e)\big|_i = v\}$*. The elements of* $Cut \cup \{\top\}$ *are called* contexts*.*

As for every $x \in V \cup E \cup Cut$ *we have exactly one context* $c \in Cut \cup \{\top\}$ *with* $x \in area(c)$*, we can write* $c = area^{-1}(x)$ *for every* $x \in area(c)$*, or even more simple and suggestive:* $c = ctx(x)$*.*

In particular the empty graph, i.e. the empty sheet of assertion, exists. Its form is $\mathfrak{G}_\emptyset := (\emptyset, \emptyset, \emptyset, \top, \emptyset, \emptyset)$.

A context c of a relational graph with cuts may contain other cuts d in its area (i.e. $d \in area(c)$), which in turn may contain further cuts, etc. It has to be expected that this idea induces an order $\leq$ on the contexts which should be a tree, having the sheet of assertion $\top$ as greatest element. The naive definition of $\leq$ is to define $c < d$ iff c is 'deeper nested' than d. The next definition is the mathematical implementation of this naive idea. Furthermore the definition extends this idea to the set of vertices and edges.

Definition 2.2 (Ordering on Contexts).

Let $(V, E, \nu, \top, Cut, area)$ *be a relational graph with cuts. We define a mapping* $\beta : V \cup E \cup Cut \cup \{\top\} \to Cut \cup \{\top\}$ *by*

$$\beta(x) := \begin{cases} x \text{ for } x \in Cut \cup \{\top\} \\ ctx(x) \text{ for } x \in V \cup E \end{cases},$$

and we set

$$x_1 \leq x_2 \;:\Longleftrightarrow\; \exists n \in \mathbb{N}_0 . \beta(x_1) \in area^n(\beta(x_2))$$

for $x_1, x_2 \in V \cup E \cup Cut \cup \{\top\}$*.*

To avoid misunderstandings, let

$$x < y :\Longleftrightarrow x \leq y \wedge y \not\leq x \qquad \text{and} \qquad x \lneq y :\Longleftrightarrow x \leq y \wedge y \neq x$$

For $c \in Cut \cup \{\top\}$, we set furthermore

$$\leq[c] := \{x \in V \cup E \cup Cut \cup \{\top\} \mid x \leq c\}$$
$$\lneq[c] := \{x \in V \cup E \cup Cut \cup \{\top\} \mid x \lneq c\}$$

In order to get an impression for this definition, have a look at Fig. 2.1 on p. 33 and Fig. 2.3 on p. 34, where we have sketched two examples for concept graphs (i.e. relational graphs which are labelled with names), and on Fig. 2.2 and Fig. 2.4, where the relation $\leq$ for these graphs is presented.

Although it has to be expected, it is not immediately clear that $\leq$ is a quasiorder. To prove this, we start with deriving some basic properties for the relation $\leq$.

Lemma 2.3 (Order Ideals Generated by Contexts).

Let $(V, E, \nu, \top, Cut, area)$ be a relational graph with cuts. Then we have for $c \in Cut \cup \{\top\}$:

$$\leq[c] = \bigcup\{area^n(c) \mid n \in \mathbb{N}_0\} \text{ , and}$$
$$\lneq[c] = \bigcup\{area^n(c) \mid n \in \mathbb{N}\} \text{ .}$$

For $c_1, c_2 \in Cut \cup \{\top\}$ we have the following implication:

$$c_1 \in \lneq[c_2] \Longrightarrow \leq[c_1] \subseteq \lneq[c_2].$$

Proof: We have

$$\begin{aligned}\leq[c] &= \{x \in V \cup E \cup Cut \cup \{\top\} \mid \exists n \in \mathbb{N}_0 . \beta(x) \in area^n(c)\} \\ &= \{k \in V \cup E \mid \exists n \in \mathbb{N}_0 . ctx(k) \in area^n(c)\} \cup \\ &\quad \{d \in Cut \cup \{\top\} \mid \exists n \in \mathbb{N}_0 . d \in area^n(c)\} \\ &= \{k \in V \cup E \mid \exists n \in \mathbb{N}_0 . k \in area^{n+1}(c)\} \cup \\ &\quad \{d \in Cut \cup \{\top\} \mid \exists n \in \mathbb{N}_0 . d \in area^n(c)\} \\ &= \bigcup_{n \in \mathbb{N}_0} area^n(c)\end{aligned}$$

Condition c) for *area* in Def. 2.1 yields that we have $\lneq[c] = \bigcup_{n \in \mathbb{N}} area^n(c)$ too. Now let $c_1 \in \lneq[c_2]$, i.e. it exists a $k \in \mathbb{N}$ with $c_1 \in area^k(c_2)$. This yields $\{c_1\} \subseteq area^k(c_2)$, and we conclude

$$\leq[c_1] = \bigcup_{n \in \mathbb{N}_0} area^n(c_1) \subseteq \bigcup_{n \in \mathbb{N}_0} area^{n+k}(c_2) \subseteq \bigcup_{n \in \mathbb{N}} area^n(c_2) = \lneq[c_2] \qquad \Box$$

The next lemma and corollary provide the main properties for the relation $<$. Lem. 2.4 states that cuts may be nested, but they do not intersect each other.

Lemma 2.4 (Relations between Order Ideals).

For a relational hypergraphs with cuts $(V, E, \nu, \top, Cut, area)$ and two contexts $c_1 \neq c_2$, exactly one of the following conditions holds:

$$i)\quad \leq[c_1] \subseteq \lneq[c_2] \qquad ii)\quad \leq[c_2] \subseteq \lneq[c_1] \qquad iii)\quad \leq[c_1] \cap \leq[c_2] = \emptyset$$

Proof: It is quite evident that neither i) and iii) nor ii) and iii) can hold simultaneously. Suppose that i) and ii) hold. We get $\leq[c_1] \subseteq \lneq[c_2] \subseteq \lneq[c_1]$, hence $c_1 \in \lneq[c_1]$, in contradiction to c) for *area* in Def. 2.1 and to Lem. 2.3. Now it is sufficient to prove the following: If iii) is not satisfied, then i) or ii) holds. So we assume that iii) does not hold. Then we have

$$\begin{aligned}
\emptyset \neq & \leq[c_1] \cap \leq[c_2] \\
= & (\{c_1\} \cup \lneq[c_1]) \cap (\{c_2\} \cup \lneq[c_2]) \\
= & (\{c_1\} \cap \{c_2\}) \cup (\{c_1\} \cap \lneq[c_2]) \cup (\{c_2\} \cap \lneq[c_1]) \cup (\lneq[c_1] \cap \lneq[c_2])
\end{aligned}$$

From $c_1 \neq c_2$ we conclude $\{c_1\} \cap \{c_2\} = \emptyset$. If $\{c_1\} \cap \lneq[c_2] \neq \emptyset$ holds, i.e. $c_1 \in \lneq[c_2]$, Lem. 2.3 yields i). Analogously follows ii) from $\{c_2\} \cap \lneq[c_1] \neq \emptyset$. So it remains to consider the case $\lneq[c_1] \cap \lneq[c_2] \neq \emptyset$. For this case, we choose $x \in area^m(c_1) \cap area^n(c_2)$ such that $n + m$ is minimal. We distinguish the following four cases:

- $m = 1 = n$: This yields $x \in area(c_1) \cap area(c_2)$ in contradiction to $c_1 \neq c_2$ and condition a) of Def. 2.1.
- $m = 1, n > 1$: Let $c_2{}' \in area^{n-1}(c_2)$ such that $x \in area(c_2{}')$. From a) of Def. 2.1 and $x \in area(c_1) \cap area(c_2{}')$ we conclude $c_1 = c_2{}'$. Hence $c_1 \in area^{n-1}(c_2)$ holds, and we get $c_1 \cup \lneq[c_1] \subseteq \lneq[c_2]$, i.e., condition i).
- $m > 1, n = 1$: From this, we conclude ii) analogously to the last case.
- $m > 1, n > 1$: Let $c_1{}' \in area^{m-1}(c_1)$ such that $x \in area(c_1{}')$, and let $c_2{}' \in area^{n-1}(c_2)$ such that $x \in area(c_2{}')$. We get $area(c_1{}') \cap area(c_2{}') \neq \emptyset$, hence $c_1{}' = c_2{}'$. This yields $area^{m-1}(c_1) \cap area^{n-1}(c_2) \neq \emptyset$, in contradiction to the minimality of $m + n$. □

Now we are prepared to show that $\leq$ is indeed a quasiorder and a tree on the set of contexts.

Corollary 2.5 ($\leq$ Induces a Tree on the Contexts).

For a relational graph with cuts $(V, E, \nu, \top, Cut, area)$, $\leq$ is a quasiorder. Furthermore, $\leq|_{Cut \cup \{\top\}}$ is an order on $Cut \cup \{\top\}$ which is a tree with the sheet of assertion $\top$ as greatest element.

Proof: We have

$$x \leq y \iff \beta(x) \leq \beta(y) \iff \leq[\beta(x)] \subseteq \leq[\beta(y)]$$

Hence $\leq$ is a quasiorder. Now Lem. 2.4 yields that the restriction of $\leq$ to $Cut \cup \{\top\}$ is an order which is furthermore a tree.

As $Cut \cup \{\top\}$ is finite, is contains a maximal element c. Assume that $c \neq \top$. Then condition b) for *area* in Def. 2.1 yields a d with $c \in area(d)$. From this we conclude $c < d$, a contradiction to the maximality of c. Hence $\top$ is the only maximal element of $Cut \cup \{\top\}$, i.e., $\top$ is the greatest element. □

The preceding lemmata are not surprising: The results had to be expected. But as the results, which are clear from a naive understanding of concept graphs, can be *proven*, the lemmata show that Def. 2.1 and Def. 2.2 seem to be a 'correct' mathematization of the underlying structure of concept graphs with cuts.

The ordered set of contexts $(Cut \cup \{\top\}, \leq)$ can be considered to be the 'skeleton' of a relational graph. According to Def. 2.1, each element of the set $V \cup E \cup Cut \cup \{\top\}$ is placed in exactly one context c (i.e. $x \in area(c)$). This motivates the next definition.

Definition 2.6 (Enclosing Relation).

Let $\mathfrak{G} := (V, E, \nu, \top, Cut, area)$ be a relational graph with cuts and let c be a context (i.e., $c \in Cut \cup \{\top\}$). Every element x of $V \cup E \cup Cut \cup \{\top\}$ with $x < c$ is said to be enclosed by *c, and vice versa: c is said to* enclose *x. For every element of $area(c)$, we say more specifically that it is* directly enclosed by *c.*

Similar to existential graphs, we need to distinguish whether the number of cuts which enclose a vertex, edge or cut is even or odd.

Definition 2.7 (Evenly/Oddly Enclosed, Pos./Neg. Contexts).

Let $\mathfrak{G} = (V, E, \nu, \top, Cut, area)$ be a relational graph with cuts. Let x be an element of $V \cup E \cup Cut \cup \{\top\}$ and set $n := |\{c \in Cut \mid x \in {\leq}[c]\}|$. If n is even, x is said to be evenly enclosed*, otherwise x is said to be* oddly enclosed*.*

The sheet of assertion $\top$ and each oddly enclosed cut is called a positive context*, and each an evenly enclosed cut is called* negative context*.*

Now we have the following: If k is a vertex or an edge, then k is evenly enclosed iff k is placed in a positive context, and k is oddly enclosed iff k is placed in a negative context. Thus, we will sometimes say that an element x be an element of $V \cup E \cup Cut \cup \{\top\}$ is positive enclosed if x is evenly enclosed, and that x is negative enclosed if x is oddly enclosed.

We will only consider graphs in which vertices must not be deeper nested than any relation they are incident with. This is captured by the following definition:

Definition 2.8 (Dominating Nodes).

If $ctx(e) \leq ctx(v)$ ($\Leftrightarrow e \leq v$) for every $e \in E$ and $v \in V_e$, then $\mathfrak{G}$ is said to have dominating nodes*.*

2.2 Concept Graphs with Cuts

The structure of simple concept graphs with cuts is derived from the structure of relational graphs with cuts. This is done by additionally labelling the vertices and edges with concept names and relation names, respectively, and by assigning a reference to each vertex. In particular all definitions concerning relational graphs with cuts (like Def. 2.2 or Def. 2.6) can be transferred to concept graphs with cuts.

We start with the definitions of the underlying alphabet (which – for conceptual graphs – is called the *support*, *taxonomy* or *ontology* by other authors), that is we start with an underlying set of variables, and with names for objects, concepts and relations. The variables and names will be needed both in concept graphs and in first order logic. The sets of concept names and relation names are ordered. These orders represent the conceptual ontology of the domain we consider. In contrast, all object names are defined to be incomparable[2]. But we will define that the generic marker '$*$' well be greater than each object name, because the generic marker is a 'more general name' than each object name (this will be important in the Beta-versions of the rules 'generalization' and 'specialization': In positive enclosed concept boxes, it will be allowed to replace object names by the generic marker $*$, and in negative enclosed concept boxes, it will be allowed to replace the generic marker $*$ by arbitrary object names).

Definition 2.9 (Alphabet).

1. *Let $Var := \{x_1, x_2, x_3, \ldots\}$ be a countably infinite set of signs.*[3] *The elements of Var are called* variables*. Let $*$ be another sign, the* generic marker*. Furthermore we assign to each variable $\alpha \in Var$ a new sign $*_\alpha$, an* indexed generic marker*.*
2. *An* alphabet *is a triple $\mathcal{A} := (\mathcal{G}, \mathcal{C}, \mathcal{R})$ of disjoint sets $\mathcal{G}$, $\mathcal{C}$, $\mathcal{R}$ such that*
   - *$\mathcal{G}$ is a finite set whose elements are called* object names*,*[4]
   - *$(\mathcal{C}, \leq_\mathcal{C})$ is a finite ordered set with a greatest element $\top$ whose elements are called* concept names*, and*

[2] In [81], implications between objects are considered.

[3] We will use the letters x, y, u, v for variables, too. Furthermore we will use the Greek letter 'α' to denote variables, i.e. 'α' is a metavariable.

[4] The letter $\mathcal{G}$ stands for the German word 'Gegenstände', i.e., 'objects'. This letter will recur when we define formal contexts where we have a set G of objects.

- $(\mathcal{R}, \leq_{\mathcal{R}})$ *is a family of finite ordered sets* $(\mathcal{R}_k, \leq_{\mathcal{R}_k})$, $k = 1, \ldots, n$ *(for an* $n \in \mathbb{N}$*) whose elements are called* relation names. *Let* $\doteq \in \mathcal{R}_2$ *be a special name which is called* identity.[5]

On $\mathcal{G} \dot{\cup} \{*\}$ *we define an order* $\leq_{\mathcal{G}}$ *such that* $*$ *is the greatest element* $\mathcal{G} \dot{\cup} \{*\}$*, but all elements of* $\mathcal{G}$ *are incomparable.*[6]

Next, we assign object names and concept names to vertices, and we assign relation names to edges.

Definition 2.10 (Simple Concept Graph with Cuts).

A simple concept graph with cuts and variables over the alphabet $\mathcal{A}$ *is a structure* $\mathfrak{G} := (V, E, \nu, \top, Cut, area, \kappa, \rho)$ *where*

- $(V, E, \nu, \top, Cut, area)$ *is a relational graph with cuts that has dominating nodes,*
- $\kappa : V \cup E \to \mathcal{C} \cup \mathcal{R}$ *is a mapping such that* $\kappa(V) \subseteq \mathcal{C}$, $\kappa(E) \subseteq \mathcal{R}$, *and all* $e \in E$ *with* $|e| = k$ *satisfy* $\kappa(e) \in \mathcal{R}_k$, *and*
- $\rho : V \to \mathcal{G} \dot{\cup} \{*\} \dot{\cup} \{*_\alpha \mid \alpha \in Var\}$ *is a mapping.*[7]

If additionally $\rho : V \to \mathcal{G} \cup \{*\}$ *holds, then* $\mathfrak{G}$ *is called* (existential) simple concept graph with cuts over the alphabet $\mathcal{A}$.[8] *The set of these graphs is denoted by* $CG^{\mathcal{A}}$. *If the alphabet is fixed, we sometimes write* CG. *If even* $\rho : V \to \mathcal{G}$ *holds, then* $\mathfrak{G}$ *is called* nonexistential simple concept graph with cuts over the alphabet $\mathcal{A}$.

For the set E *of edges, let* $E^{id} := \{e \in E \mid \kappa(e) = \doteq\}$ *and* $E^{nonid} := \{e \in E \mid \kappa(e) \neq \doteq\}$. *The elements of* E^{id} *are called* identity-links.

Finally set $V^* := \{v \in V \mid \rho(v) = *\}$ *and* $V^{\mathcal{G}} := \{v \in V \mid \rho(v) \in \mathcal{G}\}$. *The nodes in* V^* *are called* generic nodes.

[5] We will usually use the common symbol '$=$' instead of '$\doteq$', but as we use the symbol '$=$' in the meta-language, too, sometimes it will be better to use the symbol '$\doteq$' in order to distinguish it from the meta-level '$=$'.

[6] This ordering is adopted from Prediger and can be found in other papers, too. But we want to stress that other orderings are possible, too. In [80], Wille presents an approach where even the objects are ordered. It turns out that, in this approach, it is convenient to place variables and the generic marker *below* (instead above) all object names.

[7] In [45] Prediger assigned *sets* of objects instead of *single* objects to vertices (i.e. in [45] we have $\rho : V \to \mathfrak{P}(\mathcal{G}) \dot{\cup} \{*\}$ instead of $\rho : V \to \mathcal{G} \dot{\cup} \{*\}$). For concept graphs with cuts, is not immediately clear what the meaning of a vertex is which is enclosed by a cut and which contains more than one object. For this reason ρ assigns single objects to vertices.

[8] We will use the term 'existential concept graph', too. Please do not mistake existential concept graphs with the existential graphs of Peirce.

Every mapping on the vertices can naturally be extended to the edges. In particular, we extend ρ to the edges via $\rho(e) := (\rho(v_1), \ldots, \rho(v_n))$ for $e \in E$ with $\nu(e) = (v_1, \ldots, v_n)$.

In the rest of this treatise, we will mainly talk about (existential) simple concept graphs with cuts over an alphabet $\mathcal{A}$ which will be considered to be fixed, hence we will call them *concept graphs with cuts* or *concept graphs* for short.

The mathematical definitions make up an exact and solid foundation for concept graphs as syntactical constructs which can serve as a precise reference and a basis for mathematical proofs on concept graphs. But in order to work with concept graphs, their mathematical representations are too clumsy and too difficult to handle. Hence one may prefer the well known graphical representations for conceptual graphs. As we added cuts as new syntactical elements to the graphs, we have to explain how concept graphs with cuts are drawn. This shall be done now.

Usually vertices are as drawn as small rectangles. Inside the rectangle for a vertex v, we write first the concept name $\kappa(v)$ and then the reference $\rho(v)$, separated by a colon. These rectangles are called *concept boxes.*[9] This graphical notation is used in continuous text, too. e.g. we will write 'let $v = \boxed{P : g}$' instead of 'let v be a vertex with $\kappa(v) = P \in \mathcal{C}$ and $\rho(v) = g \in \mathcal{G}$'). An edge *ee* is drawn as a small oval with its relation name $\kappa(e)$ in it. These ovals are called *relation ovals.* For an edge $e = (v_1, \ldots, v_n)$, each concept box of the incident vertices $v_1, \ldots, v_n$ is connected by a line to the relation oval of e. These lines are numbered $1, \ldots, n$. If it cannot be misunderstood, this numbering is often omitted. There may be graphs such that its lines cannot be drawn without their crossing one another. In order to distinguish such lines from each other, Peirce introduced a device he called a 'bridge' or 'frog' (see [52], p. 55). But, except for bridges between lines, all the boxes, ovals, and lines of a graph must not intersect. Nearly all graphs which occur in applications do not need bridges (i.e. they are planar). Finally, a cut is drawn as a curve (usually an oval) which exactly contains in its inner space all the concept boxes, ovals, and curves of the vertices, edges, and other cuts, resp., which the cut encloses (not necessarily directly). To distinguish the curves of cuts from relation ovals, they are drawn in a different manner.

[9] Other authors use the term 'concepts' or 'conceptual instances' for concept boxes. We do not adopt these terms because in the framework of contextual logic, concepts are units of thought which consist of an extension and an intension (and we will need the term 'concept' for this understanding). Concepts should not be mixed up with objects which *belong* to a concept (i.e. to its extension). Therefore the term 'concept' should not denote something like concept boxes which refer to concept names and object names and which therefore are (atomar) judgements.

In this treatise, they are drawn bold.[10] Another possibility is to draw them in a different colour, e.g. red. The curve of a cut may not intersect any other curves, ovals or concept boxes, but it may intersect lines which connect relation ovals and concept boxes.

To illustrate these agreements, consider the following graph over the alphabet $\mathcal{A} := (\emptyset, \{\text{CAT, ANIMAL}, \top\}, \{\text{cute}, \doteq\})$ in its mathematical form:

$$\begin{aligned}\mathfrak{G} := (&\{v_1, v_2\}, \{e_1, e_2\}, \{(e_1, (v_1, v_2)), (e_2, (v_2))\}, \top, \{c_1, c_2\},\\ &\{(\top, \{c_1\}), (c_1, \{v_1, c_2\}), (c_2, \{v_2, e_1, e_2\})\},\\ &\{(v_1, \text{CAT}), (v_2, \text{ANIMAL}), (e_1, \doteq), (e_2, \text{cute})\}, \{(v_1, *), (v_2, *)\})\end{aligned}$$

In Fig. 2.1, we give one (possible) diagram for this graph. The indices v_1, v_2, e_1, e_2, c_1, c_2 do not belong to the diagram. They are added to make the translation from $\mathfrak{G}$ to the diagram more transparent. Now one can see that the intuitive meaning of the graph is 'it is not true that there is a cat which is not a cute animal', i.e. 'every cat is a cute animal'. This will be worked out in Chap. 9. In Fig. 2.2, we provide an informal way to sketch the quasiorder $\leq$ on $\mathfrak{G}$.

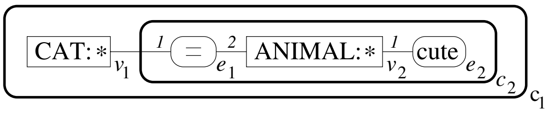


Fig. 2.1. A diagram for $\mathfrak{G}$

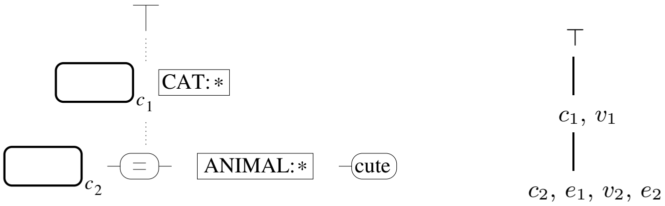


Fig. 2.2. The quasiorder $\leq$ for $\mathfrak{G}$

[10] Unfortunately, this does not correlate to the way how existential graphs are usually drawn. In existential graphs, the *lines of identity* are usually drawn bold, and the cuts are usually drawn normal. See for example the Sect. 1.1, [42] or [52].

Another example is the following graph:

$$\begin{aligned}\mathfrak{G}' := (&\{v_1, v_2\},\ \{e_1, e_2, e_3\},\ \{(e_1, (v_1)), (e_2, (v_1, v_2)), (e_3, (v_1, v_2))\}, \top,\\ &\{c_1, c_2, c_3\},\ \{(\top, \{v_1, c_1, c_2\}), (c_1, \{e_1\}), (c_2, \{v_2, e_2, c_3\}), (c_3, \{e_3\})\},\\ &\{(v_1, \text{CAT}), (v_2, \text{LASAGNE}), (e_1, \text{slim}), (e_2, \text{see}), (e_3, \text{eat})\},\\ &\{(v_1, *), (v_2, *)\})\end{aligned}$$

One diagram for $\mathfrak{G}'$ is provided in Fig. 2.3. With this diagram it is easy to see that the meaning of $\mathfrak{G}'$ is 'there is a cat which is not slim and which eats every lasagne it sees'. In the context of comics, this graph evaluates to be true (a cat fulfilling the graph is Garfield). The quasiorder $\leq$ is sketched in Fig. 2.4.

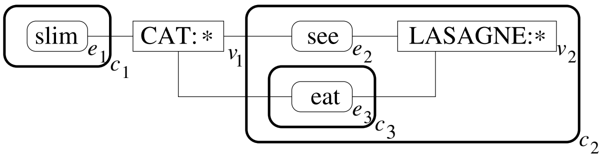


Fig. 2.3. A diagram for $\mathfrak{G}'$

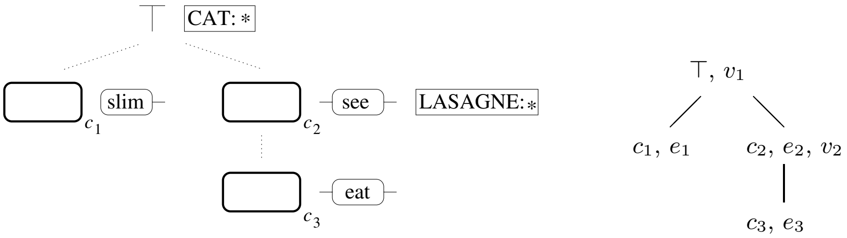
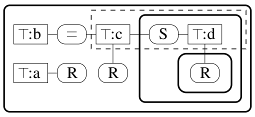


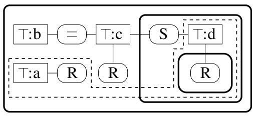
Fig. 2.4. The quasiorder $\leq$ for $\mathfrak{G}'$

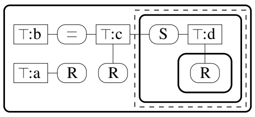
For the further treatise, especially for the calculus, the notion of a *subgraph* is needed. We distinguish between *subgraphs* and *closed subgraphs.* Informally spoken, a *subgraph* is a part of a graph placed in a context c such that

- if the subgraph contains a cut d, then it contains all what is written inside d, i.e. $\lesssim[d]$,
- every element of the subgraph is enclosed by another cut of the subgraph or by c, and
- if the subgraph contains an edge, then it contains all vertices which are incident with the edge.

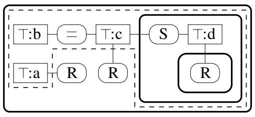
Furthermore, if for each vertex of a subgraph all incident edges are part of the subgraph, too, then the subgraph is called a *closed subgraph*. Before giving a formal definition for this, we provide some examples.

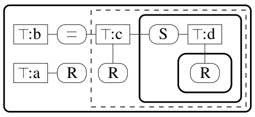




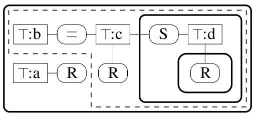


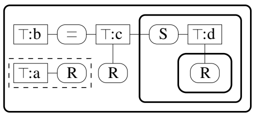
In the first example, the marked substructure contains a cut d, but it does not contain all what is written inside d. In the second example, the concept box $\boxed{\top : d}$ is not enclosed by any context of the substructure. In the last example, the substructure contains the edge with the relation name S, but is does not contain the incident vertex $\boxed{\top : c}$. Hence, in none of the examples above the marked substructure is a subgraph.





These marked substructures are subgraphs in the outermost cut which are not closed. Note that subgraphs are not necessarily connected.





In the last two examples, the marked substructures are closed subgraphs in the outermost cut.

The notion of a *subgraph* will become precise through the following definition:

Definition 2.11 (Subgraphs).

Let $\mathfrak{G} := (V, E, \nu, \top, Cut, area, \kappa, \rho)$ *be a concept graph with cuts. The graph* $\mathfrak{G}' := (V', E', \nu', \top', Cut', area', \kappa', \rho')$ *is called a* subgraph of $\mathfrak{G}$ in the context $\top'$ *if*

- $V' \subseteq V$, $E' \subseteq E$, $Cut' \subseteq Cut$ *and* $\top' \in Cut \cup \{\top\}$,
- *the mapping* ν' *is the restriction of* ν *to* E' *(and therefore well defined),*
- *the mappings* κ' *and* ρ' *are the restriction of* κ *to* $V' \cup E'$ *and* ρ *to* V'*, respectively,*
- $area'(\top') = area(\top') \cap (V' \cup E' \cup Cut')$ *and* $area'(c') = area(c')$ *for each* $c' \in Cut'$,
- $ctx(x) \in Cut' \cup \{\top'\}$ *for each* $x \in V' \cup E' \cup Cut'$ *, and*
- $V_{e'} \subseteq V'$ *for each edge* $e' \in E'$.

We write: $\mathfrak{G}' \subseteq \mathfrak{G}$ *and* $area^{-1}(\mathfrak{G}') = \top'$ *resp.* $ctx(\mathfrak{G}') = \top'$.

If we additionally have $E_{v'} \in E'$ *for each vertex* $v' \in V'$*, then* $\mathfrak{G}'$ *is called* closed subgraph of $\mathfrak{G}$ in the context $\top'$.

Note that the condition $area'(c') = area(c')$ for $c' \in Cut'$ yields that we have $\lneqq[c'] \subseteq V' \cup E' \cup Cut'$ for each $c' \in Cut'$. In particular, Cut' is an ideal in $Cut \cup \{\top\}$ (i.e. we have that $c' \in Cut'$ and $d \leq c'$ imply $d \in Cut'$), but in general, $Cut' \cup \{\top'\}$ is *not* an ideal in $Cut \cup \{\top\}$ (there may be contexts $d \in area(\top')$ which are not a element of $Cut' \cup \{\top'\}$).

As subgraphs are, similar to elements of $E \cup V \cup Cut \cup \{\top\}$, placed in contexts, we can apply Def. 2.6 and Def. 2.7 to subgraphs as well. Hence we say that subgraphs are (directly) enclosed by cuts, and we distinguish whether they are evenly or oddly enclosed.

Finally we want to point out that subgraphs are indeed concept graphs.

We need to define when two concept graphs are isomorphic. Though this definition is technical and not too short, it is rather straightforward.

Definition 2.12 (Isomorphism).

Let $\mathfrak{G}_i := (V_i, E_i, \nu_i, \top_i, Cut_i, area_i, \kappa_i, \rho_i)$, $i = 1, 2$ *be two simple concept graphs with cuts. Then* $f = f_V \dot\cup f_E \dot\cup f_{Cut}$ *is called* isomorphism*, if*

- $f_V : V_1 \to V_2$ *is bijective,*
- $f_E : E_1 \to E_2$ *is bijective,*
- $f_{Cut} : Cut_1 \cup \{\top_1\} \to Cut_2 \cup \{\top_2\}$ *is bijective with* $f_{Cut}(\top_1) = \top_2$,

such that the following conditions hold:

- *Each* $e = (v_1, \ldots, v_n) \in E_1$ *satisfies* $f_E(v_1, \ldots, v_n) = (f_V(v_1), \ldots, f_V(v_n))$ *(edge condition),*
- $f[area_1(c)] = area_2(f(c))$ *for each* $c \in Cut_1 \cup \{\top_1\}$ *(cut condition),* *(with* $f[area_1(c)] = \{f(k) \mid k \in area_1(c)\}$*),*
- $\rho_1(v) = \rho_2(f_V(v))$ *for all* $v \in V_1$,
- $\kappa_1(v) = \kappa_2(f_V(v))$ *for all* $v \in V_1$ *, and*
- $\kappa_1(e) = \kappa_2(f_E(e))$ *for all* $e \in E_1$.

Furthermore, a notation of a partial isomorphism is needed. The reason for this is as follows: Most of the rules in the calculus we will present modify only parts of a graph which are enclosed by a specific context. So we need to express that the starting graph and the modified graph are isomorphic except for the area of this specific context.

Definition 2.13 (Isomorphims except Cuts).

For $i = 1, 2$, let $\mathfrak{G}_i := (V_i, E_i, \nu_i, \top_i, Cut_i, area_i, \kappa_i, \rho_i)$, be simple concept graphs with cuts and $c_i \in Cut_i \cup \{\top_i\}$ contexts. Let $E_i' := \{e \in E_i \mid e \not\leq c_i\}$, $V_i' := \{v \in V_i \mid v \not\leq c_i\}$ and $Cut_i{}' := \{d \in Cut_i \cup \{\top_i\} \mid d \not< c_i\}$ for $i = 1, 2$. Then $f = f_V \dot\cup f_E \dot\cup f_{Cut}$ is called isomorphism except for $c_1 \in Cut_1 \cup \{\top_1\}$ and $c_2 \in Cut_2 \cup \{\top_2\}$ *if*

- $f_V : V_1' \to V_2'$ *is bijective,*
- $f_E : E_1' \to E_2'$ *is bijective,*
- $f_{Cut} : Cut_1{}' \to Cut_2{}'$ *is bijective with* $f_{Cut}(\top_1) = \top_2$,

such that the following conditions hold:

- *For* $e = (v_1, \ldots, v_n) \in E_1'$ *it holds* $f_E(v_1, \ldots, v_n) = (f_V(v_1), \ldots, f_V(v_n))$ *(edge condition),*
- $f[area(c)] = area'(f(c))$ *for each* $c \in (Cut_1{}')$ *(cut condition),*
- $\rho_1(v) = \rho_2(f_V(v))$ *holds for all* $v \in V_1'$,
- $\kappa_1(v) = \kappa_2(f_V(v))$ *for all* $v \in V_1'$ *, and*
- $\kappa_1(e) = \kappa_2(f_E(e))$ *for all* $e \in E_1'$.

Please note that we have defined $Cut_i{}' := \{d \in Cut_i \cup \{\top_i\} \mid d \not< c_i\}$ instead of $Cut_i{}' := \{d \in Cut_i \cup \{\top_i\} \mid d \not\leq c_i\}$. This yields that we have $c_i \in Cut_i'$ for $i = 1, 2$ (but not $v_i \in V_i'$ for $v_i \in area(c_i) \cap V_i$, $i = 1, 2$ and not $e_i \in E_i'$ for $e_i \in area(c_i) \cap E_i$, $i = 1, 2$).

To see examples for Def. 2.13, look at the example for the iteration-rule in Alpha on p. 52 and at Example 10.2 on p. 110 for the iteration-rule in Beta. In both examples, a subgraph of a graph is iterated into a cut. The starting graph and the modified graph are isomorphic except for the cut the subgraph is iterated into.

Finally, we have the possibility to write several graphs side by side. This is a common operation on concept graphs which is called *juxtaposition*. Mathematically, it is the disjoint[11] union of a set of graphs, which is captured by the next definition.

Definition 2.14 (Juxtaposition).

Let $n \in \mathbb{N}_0$ and $\mathfrak{G}_i := (V_i, E_i, \nu_i, \top_i, Cut_i, area_i, \kappa_i, \rho_i)$ be a concept graph with cuts for $i = 1, \ldots, n$. The juxtaposition of the $\mathfrak{G}_i$ *is defined to be the following concept graph $\mathfrak{G} := (V, E, \nu, \top, Cut, area, \kappa, \rho)$:*

[11] In order to ensure that the graphs are disjoint, a simple technical trick – all sets of the ith graph are indexed by i – is applied in the definition.

- $V := \bigcup_{i=1,\ldots,n} V_i \times \{i\}$,
- $E := \bigcup_{i=1,\ldots,n} E_i \times \{i\}$,
- *for* $e = (v_1, \ldots, v_n) \in E$, *let* $\nu((e,i)) := ((v_1,i), \ldots, (v_n,i))$,
- $\top$ *arbitrary element,*
- $Cut := \bigcup_{i=1,\ldots,n} Cut_i \times \{i\}$,
- *area is defined as follows:* $area((c,i)) = area_i(c) \times \{i\}$ *for* $c \in Cut_i$
  $area(\top) = \bigcup_{i=1,\ldots,n} area_i(\top_i) \times \{i\}$
- $\kappa(k,i) := \kappa_i(k)$ *for all* $k \in V \cup E$ *and* $i = 1, \ldots, n$, *and*
- $\rho(e,i) := \rho_i(k)$ *for all* $e \in E$ *and* $i = 1, \ldots, n$.

In the graphical notation, the juxtaposition of the $\mathfrak{G}_i$ *is simply noted by writing the graphs next to each other, i.e. we write:*

$$\mathfrak{G}_1 \ \mathfrak{G}_2 \ \ldots \ \mathfrak{G}_n$$

We want to point out that the juxtaposition of an empty set of concept graphs is allowed, too. It is easy to see that the juxtaposition of an empty set of concept graphs yields the empty concept graph.

3 Overview for Alpha

We start our investigations with *nonexistential* concept graphs with cuts, i.e., concept graphs *without* generic markers. The approach for nonexistential concept graphs with cuts is straight forward: It is based on syntax, semantics, and a calculus which is sound and complete.

As the syntax for these graphs is already defined in Chap. 2, we start by providing the semantics. The semantics we present are *contextual*, that is they are based on power context families (i.e., families of formal contexts, as the are introduced in FCA). They are similar to classical relational models as they are used in FOL, but in contrast to those, they bear intensional information, too. We will approach this subject in the part Beta of this treatise. In the part Alpha it is enough to present the basic notions of FCA and the contextual semantics.

In the following chapter we present the calculus for nonexistential concept graphs. It is based on the calculus of Charles S. Peirce for existential graphs. This calculus has five rules, and these rules are adapted for concept graphs. But the calculus for concept graphs contains additional rules. In Sect. 5.1 we present the calculus in common language as well as in a precise mathematical way. To understand what the rules are doing, the descriptions in common language are easier to understand, but these descriptions may be ambiguous. Hence the mathematical definitions are needed to clarify and to explain in a mathematically precise way the common language descriptions. In the next section we provide some examples and heuristics for the rules. In Sect. 5.3 we provide some simple theorems and show that the rules are sufficient to transform each graph into some kinds of normal forms.

In Chap. 6 we mainly show that the calculus is sound and complete with respect to the given contextual semantics. In Sect. 6.1 we start with the proof of the soundness. As the rules of the calculus are very powerful, it will turn out that this proof is not trivial. In Sect. 6.2 we construct for each graph which is not contradictory a model which fulfills the graph, and from this we immediately conclude that the calculus is complete, too. The idea for this proof is adopted from the usual way to show that propositional calculi and first order calculi are sound. We have already mentioned that the calculus contains the concept graph version of Peirce's rules for existential graphs and

F. Dau: The Logic System of Concept Graphs with Negation, LNAI 2892, pp. 39-40, 2003.

some further rules. In Sect. 6.3, we show for each of these additional rules that the calculus is strictly weakened if that rule is omitted, i.e., the new rules are necessary to get a complete calculus. In particular we see that Peirce's rules are not sufficient in the system of concept graphs with cuts, i.e., they do not form a complete calculus.

4 Semantics for Nonexistential Concept Graphs

In this chapter we provide a semantics for nonexistential graphs with cuts. In literature we find different kinds of semantics for conceptual graphs:

- a direct extensional semantics,
- a translation to FOL by the 'well-known' operator Φ,
- the game-theoretical semantics by Hintikka and
- the contextual semantics by Prediger and Wille.

The direct extensional semantics was presented in [9], but is was rarely used.

The translation to FOL is the most established semantics for conceptual graphs (see for example [1], [11], [7], [59], [60], [62], [58], [64], [65], [70]). Note that this semantics is given by translating one syntactically given structure into another syntactically given structure. For this reason, in our view this use of the term 'semantics' is not appropriate. But the classical extensional semantics of FOL-formulas (i.e., relational structures) can serve via the operator Φ as extensional semantics for conceptual graphs as well. We will re-assume and work out this approach in the Beta-Part of this treatise (in particular in Sect. 9.3).

The game-theoretical semantics of Hintikka (see [24], [62]) is closely related to the endoporeutic method of Peirce, which yields the meaning of existential graphs. In this treatise, you find more on the endoporeutic method in Sect. 9. For a further discussion we refer to [52].

The semantics which will be used is this treatise is the contextual semantics which is based on Formal Concept Analysis, as it was introduced by Wille in [75] and worked out by Prediger in [44] for concept graphs without negations.[1] Thus, the underlying structure of the models will be so-called *power context*

[1] We want to remark that only Prediger implements a strict separation between syntax (concept graphs) and semantics (power context families). For the purpose of this treatise, in particular for the proof that concept graphs with cuts and first order logic are equivalent (which will be done in the part Beta of this treatise), this separation is convenient, so we adopt the approach of Prediger. In contrast to Prediger, Wille does not separate the syntax of concept graphs: He defines concept graphs as *semantical* structures. Thus, as he treats both concept graphs

F. Dau: The Logic System of Concept Graphs with Negation, LNAI 2892, pp. 41-44, 2003.

families. It has to be argued why we do not simply use the 'classical' relational structures of first order logic in our semantics. As Wille says in [75]: ' There is a fundamental reason for associating concept(ual) graphs and Formal Concept Analysis which lies in their far-back reaching roots in philosophical logic and in their pragmatic orientation; more specifically, both together can play a substantial role in the formalization of (elementary) logic. [...] Elementary logic was understood and taught by the traditional paradigm of philosophical logic based on the three essential main functions of thinking – *concepts*, *judgements* and *conclusions*.' A formalization of this understanding of logic is an adaquate approach for knowledge representation and processing, which is a main goal of conceptual graphs and concept graphs. The formalization of concepts, particularly the understanding of a concept as a unit of thought constituted by its extension and intension, has been successfully established by Formal Concept Analysis. A crucial point is that the extension and intension of a concept can only be grasped in a fixed universe of discourse, which is formalized as a so-called *formal context*. Due to this contextual view, the sementaics which is presented here is called *contextual semantics*. In the line of the philosophical understanding of logic, concept graphs can be understood as a formalization of judgements, and the deductive procedures on concept graphs can be understood as a formalization of conclusions. For a deeper discussion on this topic, we refer to [75]. The mathematical theory of Formal Concept Analysis is worked out in [19].

If we do not use any cuts, the semantics we present works exactly like the semantics of Prediger. Thus, it can be understood as an extension of her approach to concept graphs with cuts. On the other hand, the use of cuts forces an iterated evaluation of a concept graph with cuts in a model. This iteration works similar to the endoporeutic method of Peirce, hence it is similar to the game-theoretical approach of Hintikka. In other words: The approach in this treatise can be understood as combination of the ideas of Peirce, Wille and Prediger, and Hintikka.

In concept graph *without* cuts, only the conjunction of positive information can be expressed. For this reason it was possible for Prediger to construct for each concept graph a so-called *standard model* which encodes exactly all information of the concept graph. Standard models have been an additional possibility (besides the entailment relation and the calculus) for doing reasoning with concept graphs. Even in the theory of semi-concept graphs, where negation can be expressed on the level on (semi-)concepts (see [28], [29], [30], [79],[80]), a weaker version of standard models, so-called standard power context families, can be implemented. We will come back to the no-

and power context families as semantical structures, his approach yields a closer, more direct connection between these two structures. In particular, he is able to define *concept graphs of power context families*, and, vice versa, *power context families of concept graphs*.

tion of standard models in the last chapter of the Beta-part, where Prediger's approach is adopted end extended for concept graphs without cuts.

With concept graphs with cuts, negation can be expressed on the level of judgements.[2] In particular, the DeMorgan-laws allow us to express the *disjunction* of propositions. But disjunction of propositions cannot be encoded in standard models. Thus, for concept graphs with cuts, it is unfortunately not possible to construct standard models.

Now let us recall the basic definition of Wille and Prediger.

Definition 4.1 (Formal Contexts).

A formal context *is a triple* $\mathbb{K} := (G, M, I)$*, where* G *is a set of (formal) objects,* M *is a set of (formal) attributes and* $I \subseteq G \times M$ *is an incidence-relation. A pair* (A, B) *with* $A \subseteq G$ *and* $B \subseteq M$ *is called a* formal concept of $\mathbb{K}$*, if and only if* $A = \{g \in G \mid gIn \textit{ for all } b \in B\}$ *and* $B = \{m \in M \mid aIm \textit{ for all } a \in A\}$*.* $\mathit{Ext}(A, B) := A$ *is called* extension of the concept (A, B)*, and* $\mathit{Int}(A, B) := B$ *is called* intension of the concept (A, B)*. The set of all formal concepts of a formal context* $\mathbb{K}$ *is denoted by* $\mathfrak{B}(\mathbb{K})$ *.*

Definition 4.2 (Power Context Families).

A power context family $\vec{\mathbb{K}} := (\mathbb{K}_0, \mathbb{K}_1, \mathbb{K}_2, \ldots)$ *is a family of contexts* $\mathbb{K}_k := (G_k, M_k, I_k)$ *that satisfies* $G_0 \neq \emptyset$ *and* $G_k \subseteq (G_0)^k$ *for each* $k \in \mathbb{N}$*. Then we write* $\vec{\mathbb{K}} := (G_k, M_k, I_k)_{k \in \mathbb{N}_0}$*. The elements of* G_0 *are the* objects *of* $\vec{\mathbb{K}}$*. All elements of* $\bigcup_{k \in \mathbb{N}_0} \mathfrak{B}(\mathbb{K}_k)$ *are called* concepts*. We set furthermore* $\mathfrak{R}_{\vec{\mathbb{K}}} := \bigcup_{k \in \mathbb{N}} \mathfrak{B}(\mathbb{K}_k)$*, and the elements of* $\mathfrak{R}_{\vec{\mathbb{K}}}$ *are called* relation-concepts*.*

When interpreting a concept graph in a power context family, the object names of our alphabet will be interpreted by objects, the concept names of our alphabet will be interpreted by formal concepts of the context $\mathbb{K}_0$, the relation names of arity k will be interpreted by relation-concepts of $\mathbb{K}_k$, and this interpretation has to respect the orders on the names. This motivates the following definition:

Definition 4.3 (Contextual Models).

Let $\mathcal{A} := (\mathcal{G}, \mathcal{C}, \mathcal{R})$ *be an alphabet and* $\vec{\mathbb{K}}$ *be a power context family. Then we call the disjoint union* $\lambda := \lambda_{\mathcal{G}} \dot\cup \lambda_{\mathcal{C}} \dot\cup \lambda_{\mathcal{R}}$ *of the mappings* $\lambda_{\mathcal{G}} \colon \mathcal{G} \to G_0$*,* $\lambda_{\mathcal{C}} \colon \mathcal{C} \to \underline{\mathfrak{B}}(\mathbb{K}_0)$ *and* $\lambda_{\mathcal{R}} \colon \mathcal{R} \to \mathfrak{R}_{\vec{\mathbb{K}}}$ *a* $\vec{\mathbb{K}}$-interpretation *of* $\mathcal{A}$ *if* $\lambda_{\mathcal{C}}$ *and* $\lambda_{\mathcal{R}}$ *are order-preserving, and* $\lambda_{\mathcal{C}}, \lambda_{\mathcal{R}}$ *satisfy* $\lambda_{\mathcal{C}}(\top) = \top$*,* $\lambda_{\mathcal{R}}(\mathcal{R}_k) \subseteq \underline{\mathfrak{B}}(\mathbb{K}_k)$ *for all*

[2] We have to stress that judgements are *asserted* propositions, i.e., judgements claim to be true. For this reason, the negation of a judgement cannot be considered to be a judgement again. In particular, in our approach it is possible to build graphs which are contradictory, and these graphs cannot be asserted. In this case, we speak of 'propositions' instead of 'judgements'.

$k = 1, \dots, n$, and $(g_1, g_2) \in Ext(\lambda_{\mathcal{R}}(\doteq)) \Leftrightarrow g_1 = g_2$ for all $g_1, g_2 \in G_0$. The pair $(\vec{\mathbb{K}}, \lambda)$ is called contextual model *over $\mathcal{A}$ or* contextual structure over *$\mathcal{A}$.*[3]

Recall that mappings on V can naturally be extended to mappings on E. As $\lambda_{\mathcal{G}}$ is a mapping on the set $\mathcal{G}$ of object names, it can be naturally extended to tuples of object names. In particular we set $\lambda_{\mathcal{G}}(\rho(e)) := (\lambda_{\mathcal{G}}(\rho(v_1)), \dots, \lambda_{\mathcal{G}}(\rho(v_n)))$.

Now we can define whether a concept graph is valid in a contextual structure over $\mathcal{A}$.

Definition 4.4 (Evaluation).

Let $(\vec{\mathbb{K}}, \lambda)$ be a contextual structure over $\mathcal{A}$ and let $\mathfrak{G}$ be a concept graph. We evaluate $\mathfrak{G}$ inductively over $c \in Cut \cup \{\top\}$. The evaluation of $\mathfrak{G}$ in a context c is written $(\vec{\mathbb{K}}, \lambda) \models \mathfrak{G}[c]$, and it is inductively defined as follows:

$(\vec{\mathbb{K}}, \lambda) \models \mathfrak{G}[c] :\Longleftrightarrow$

- *$\lambda_{\mathcal{G}}(\rho(v)) \in Ext(\lambda_{\mathcal{C}}(\kappa(v)))$ for each $v \in V \cap area(c)$ (vertex condition)*
- *$\lambda_{\mathcal{G}}(\rho(e)) \in Ext(\lambda_{\mathcal{R}}(\kappa(e)))$ for each $e \in E \cap area(c)$ (edge condition)*
- *$(\vec{\mathbb{K}}, \lambda) \not\models \mathfrak{G}[c']$ for each $c' \in Cut \cap area(c)$ (cut condition: iteration over $Cut \cup \{\top\}$)*

For $(\vec{\mathbb{K}}, \lambda) \models \mathfrak{G}[\top]$ we write $(\vec{\mathbb{K}}, \lambda) \models \mathfrak{G}$ and say that $\mathfrak{G}$ is valid in *$(\vec{\mathbb{K}}, \lambda)$ resp. $(\vec{\mathbb{K}}, \lambda)$* is a contextual model *for $\mathfrak{G}$.*

If we have two concept graphs $\mathfrak{G}_1$, $\mathfrak{G}_2$ such that $(\vec{\mathbb{K}}, \lambda) \models \mathfrak{G}_2$ for each contextual structure over $\mathcal{A}$ with $(\vec{\mathbb{K}}, \lambda) \models \mathfrak{G}_1$, we write $\mathfrak{G}_1 \models \mathfrak{G}_2$.

Intuitively, $(\vec{\mathbb{K}}, \lambda) \models \mathfrak{G}[c]$ can be read as $(\vec{\mathbb{K}}, \lambda) \models \mathfrak{G}\big|_{\lneq[c]}$. But generally $\lneq[c]$ is not a subgraph of $\mathfrak{G}$. Therefore this should not be understood as a precise definition.

Furthermore we want to remark that the semantics we present here does not require that the graphs have dominating nodes: Definition 4.4 works for concept graphs *without* dominating nodes as well. This will be different when we introduce the generic marker to concept graphs with cuts (see Chap. 9).

[3] The name 'contextual structure' is chosen according to the term 'relational structure' which is the name for the usual extensional models in FOL (see Def. 8.6)

5 Calculus for Nonexistential Concept Graphs

In this chapter, in the first section, we will provide a calculus for nonexistential concept graphs with cuts. In the second section we will give some examples for the rules, as well as some heuristics for their usage. Finally, in the last section, we will prove some simple meta-theorems, and we will define some types of normal-forms in which each graph can be transformed.

5.1 Calculus

In this section the calculus for non-existential concept graphs is provided. It is inspired by Peirce's calculus for EGs. Particulary, Peirce's five rules are adopted, and – except for the empty sheet of assertion – we have no axioms. For the sake of intelligibility, the whole calculus is first described using common spoken language, then the mathematical definitions are given.

Definition 5.1 (Calculus for Nonex. Concept Graphs with Cuts). *The calculus for nonexistential concept graphs with cuts over the alphabet $\mathcal{A}$ consists of the following rules:*

- **erasure**

  In positive contexts, any directly enclosed edge, isolated vertex, and closed subgraph may be erased.
- **insertion**

  In negative contexts, any directly enclosed edge, isolated vertex, and closed subgraph may be inserted.
- **iteration**

  Let $\mathfrak{G}_0 := (V_0, E_0, \nu_0, \top_0, Cut_0, area_0, \kappa_0, \rho_0)$ be a (not necessarily closed) subgraph of $\mathfrak{G}$ and let $c \leq ctx(\mathfrak{G}_0)$ be a context such that $c \notin Cut_0$. Then a copy of $\mathfrak{G}_0$ may be inserted into c.
- **deiteration**

  If $\mathfrak{G}_0$ is a subgraph of $\mathfrak{G}$ which could have been inserted by rule of iteration, then it may be erased.

F. Dau: The Logic System of Concept Graphs with Negation, LNAI 2892, pp. 45-62, 2003.

- **double cut**

  Double cuts (cuts c_1, c_2 with $area(c_1) = \{c_2\}$) may be inserted or erased.
- **generalization**

  For evenly enclosed vertices and edges, the concept names respectively relation names may be generalized.
- **specialization**

  For oddly enclosed vertices and edges, the concept names respectively relation names may be specialized.
- **isomorphism**

  A graph may be substituted by an isomorphic copy of itself.
- **exchanging references**

  Let $e \in E^{id}$ be an identity link with $\rho(e\big|_1) = g_1$, $\rho(e\big|_2) = g_2$ and $ctx(e) = ctx(e\big|_1) = ctx(e\big|_2)$. Then the references of v_1 and v_2 may be exchanged.
- **merging two vertices**

  Let $v_1 = \boxed{P : g}$ and $v_2 = \boxed{\top : g}$ be two vertices with $ctx(v_1) \geq ctx(v_2)$. Then v_2 may be merged into v_1 (i.e., v_2 is erased and, for every edge $e \in E$, $e|_i = v_2$ is replaced by $e|_i = v_1$).
- **splitting a vertex**

  Let $g \in \mathcal{G}$. Let $v_1 = \boxed{P : g}$ be a vertex, incident with relation edges $e_1, \ldots, e_n$, and let c be a context with $ctx(v_1) \geq c \geq ctx(e_1), \ldots, ctx(e_n)$. Then the following may be done: In c, a new vertex $v_2 = \boxed{\top : g}$ is inserted. On $e_1, \ldots, e_n$, arbitrary occurences of v_1 may be substituted by v_2.
- **$\top$-erasure**

  For each object name g, an isolated vertex $\boxed{\top : g}$ may be erased from arbitrary contexts.
- **$\top$-insertion**

  For each object name g, an isolated vertex $\boxed{\top : g}$ may be inserted into arbitrary contexts.
- **identity-erasure**

  Let $g \in \mathcal{G}$, let $v_1 = \boxed{P_1 : g}$ and $v_2 = \boxed{P_2 : g}$ be two vertices. Then any identity-link between v_1 and v_2 may be erased.
- **identity-insertion**

  Let $g \in \mathcal{G}$, let $v_1 = \boxed{P_1 : g}$, $v_2 = \boxed{P_2 : g}$ be two vertices in contexts c_1, c_2, resp. and let $c \leq c_1, c_2$ be a context. Then an identity-link between v_1 and v_2 may be inserted into c.

These rules have to be written down mathematically. Here are the appropriate mathematical definitions:

- **erasure and insertion, $\top$-erasure and $\top$-insertion, identity-erasure and identity-insertion**

  We first provide general definitions for inserting and erasing vertices, edges and closed subgraphs.

  Let $e \in E$ be an edge with $ctx(e) = c$.

  Let $\mathfrak{G}^{(e)} := (V^{(e)}, E^{(e)}, \nu^{(e)}, \top^{(e)}, Cut^{(e)}, area^{(e)}, \kappa^{(e)}, \rho^{(e)})$ be the following graph:

  - $V^{(e)} := V$
  - $E^{(e)} := E \backslash \{e\}$
  - $\nu^{(e)} := \nu|_{E^{(e)}}$
  - $\top^{(e)} := \top$
  - $Cut^{(e)} := Cut$
  - $area^{(e)}(d) := area(d) \backslash \{e\}$ for all $d \in Cut^{(e)} \cup \{\top^{(e)}\}$
  - $\kappa^{(e)} := \kappa|_{V^{(e)} \cup E^{(e)}}$
  - $\rho^{(e)} := \rho$

  Then we say that $\mathfrak{G}^{(e)}$ is derived from $\mathfrak{G}$ by *erasing the edge e from the context c*, and $\mathfrak{G}$ is derived from $\mathfrak{G}^{(e)}$ by *inserting the edge e into the context c*.

  Let $v \in V$ be an isolated vertex (i.e., $E_v = \emptyset$) with $ctx(v) = c$.

  Let $\mathfrak{G}^{(v)} := (V^{(v)}, E^{(v)}, \nu^{(v)}, \top^{(v)}, Cut^{(v)}, area^{(v)}, \kappa^{(v)}, \rho^{(v)})$ be the following graph:

  - $V^{(v)} := V \backslash \{v\}$
  - $E^{(v)} := E$
  - $\nu^{(v)} := \nu$
  - $\top^{(v)} := \top$
  - $Cut^{(v)} := Cut$
  - $area^{(v)}(d) := area(d) \backslash \{v\}$ for all $d \in Cut^{(v)} \cup \{\top^{(v)}\}$
  - $\kappa^{(v)} := \kappa|_{V^{(v)} \cup E^{(v)}}$
  - $\rho^{(v)} := \rho|_{V^{(v)}}$

  Then we say that $\mathfrak{G}^{(v)}$ is derived from $\mathfrak{G}$ by *erasing the vertex v from the context c*, and $\mathfrak{G}$ is derived from $\mathfrak{G}^{(v)}$ by *inserting the vertex v into the context c*.

Let $\mathfrak{G} := (V, E, \nu, \top, Cut, area, \kappa, \rho)$ be a concept graph which contains the closed subgraph $\mathfrak{G}_0 := (V_0, E_0, \nu_0, \top_0, Cut_0, area_0, \kappa_0, \rho_0)$.

Let $\mathfrak{G}' := (V', E', \nu', \top', Cut', area', \kappa', \rho')$ be the following graph:

- $V' := V \backslash V_0$
- $E' := E \backslash E_0$
- $\nu' := \nu|_{E'}$
- $\top' := \top$
- $Cut' := Cut \backslash Cut_0$
- $area'(d) := \begin{cases} area(d) & d \neq \top_0 \\ area(d) \backslash (V_0 \cup E_0 \cup Cut_0) & d = \top_0 \end{cases}$.
- $\kappa' := \kappa|_{V' \cup E'}$
- $\rho' := \rho|_{V'}$

Then we say that $\mathfrak{G}'$ is derived from $\mathfrak{G}$ by *erasing the subgraph* $\mathfrak{G}_0$ *from the context* $\top_0$, and $\mathfrak{G}$ is derived from $\mathfrak{G}'$ by *inserting the graph* $\mathfrak{G}_0$ *into the context* $\top_0$.

Now the rules erasure and insertion, $\top$-erasure and $\top$-insertion, identity-erasure and identity-insertion are restrictions of the general definitions above:

- **erasure and insertion**

  Let $\mathfrak{G}$ be a concept graph and let k be an isolated vertex, an edge or a subgraph of $\mathfrak{G}$ with $c := ctx(k)$, and let $\mathfrak{G}'$ be obtained from $\mathfrak{G}$ by erasing k from the context c. If c is positive, then $\mathfrak{G}'$ is derived from $\mathfrak{G}$ by *erasing k from a positive context*, and if c is negative, than $\mathfrak{G}'$ is derived from $\mathfrak{G}$ by *inserting k into a negative context.*

- **$\top$-erasure and $\top$-insertion**

  Let $\mathfrak{G} := (V, E, \nu, \top, Cut, area, \kappa, \rho)$ be a concept graph and let $v \in V$ be a vertex with $E_v = \emptyset$ and $\kappa(v) = \top$, and let $g = \rho(v)$. Let $\mathfrak{G}'$ be obtained from $\mathfrak{G}$ by erasing v from the context $ctx(v)$. Then $\mathfrak{G}'$ is derived from $\mathfrak{G}$ by *erasing the isolated vertex* $v = \boxed{\top : g}$, and $\mathfrak{G}$ is derived from $\mathfrak{G}'$ by *inserting the isolated vertex* $v = \boxed{\top : g}$.

- **identity-erasure and identity-insertion**

  Let $\mathfrak{G} := (V, E, \nu, \top, Cut, area, \kappa, \rho)$ be a concept graph and let $e = (v_1, v_2) \in E^{(2)}$ be an edge with $\kappa(e) = \doteq$ and $\rho(v_1) = \rho(v_2) = g \in \mathcal{G}$. Let $\mathfrak{G}'$ be obtained from $\mathfrak{G}$ by erasing e from the context $ctx(e)$. Then $\mathfrak{G}'$ is derived from $\mathfrak{G}$ by *erasing an identity link between* $v_1 = \boxed{P_1 : g}$ *and* $v_2 = \boxed{P_1 : g}$, and $\mathfrak{G}$ is derived from $\mathfrak{G}'$ by *inserting an identity link between* $v_1 = \boxed{P_1 : g}$ *and* $v_2 = \boxed{P_1 : g}$.

- **iteration and deiteration**

  Let $\mathfrak{G} := (V, E, \nu, \top, Cut, area, \kappa, \rho)$ is a concept graph with the (not necessarily closed) subgraph $\mathfrak{G}_0 := (V_0, E_0, \nu_0, \top_0, Cut_0, area_0, \kappa_0, \rho_0)$ and let c be a context with $c \notin Cut_0$.

  Let $\mathfrak{G}' := (V', E', \nu', \top', Cut', area', \kappa', \rho')$ be the following graph:

  - $V' := V \times \{1\} \;\cup\; V_0 \times \{2\}$
  - $E' := E \times \{1\} \;\cup\; E_0 \times \{2\}$
  - $\nu'((e,i)) = ((v_1,i), \ldots, (v_n,i))$ for $(e,i) \in E'$ and $\nu(e) = (v_1, \ldots, v_n)$
  - $\top' := \top$
  - $Cut' := Cut \times \{1\} \;\cup\; Cut_0 \times \{2\}$
  - $area'$ is defined as follows:

    for $(d,i) \in Cut'$ and $d \neq c$ let $area'((d,i)) := area(d) \times \{i\}$ and let

    $area'((c,1)) := area(c) \times \{1\} \;\cup\; area_0(\top_0) \times \{2\}$
  - $\kappa'((k,i)) := \kappa(k)$ for all $(k,i) \in V' \cup E'$
  - $\rho'((v,i)) = \rho(v)$ for all $(v,i) \in V'$

  Then we say that $\mathfrak{G}'$ is derived from $\mathfrak{G}$ by *iterating the subgraph* $\mathfrak{G}_0$ *into the context* c and $\mathfrak{G}$ is derived from $\mathfrak{G}'$ by *deiterating the subgraph* $\mathfrak{G}_0$ *from the context* c.

- **double cuts**

  Let $\mathfrak{G} := (V, E, \nu, \top, Cut, area, \kappa, \rho)$ be a concept graph and $c_1, c_2 \in Cut$ such that $area(c_1) = \{c_2\}$. Let $c_0 := ctx(c_1)$ (i.e., $c_1 \in area(c_0)$) and set $\mathfrak{G}' := (V, E, \nu, \top, Cut', area', \kappa, \rho)$ with

  - $Cut' := Cut \backslash \{c_1, c_2\}$
  - $area'(d) := \begin{cases} area(d) \text{ for } d \neq c_0 \\ area(c_0) \cup area(c_2) \text{ for } d = c_0 \end{cases}.$

  Then we say that $\mathfrak{G}'$ is derived from $\mathfrak{G}$ by *erasing the double cuts* c_1, c_2 and $\mathfrak{G}$ is derived from $\mathfrak{G}'$ by *inserting the double cuts* c_1, c_2.

- **isomorphism**

  If $\mathfrak{G}$ and $\mathfrak{G}'$ are isomorphic, then we say that $\mathfrak{G}'$ is derived from $\mathfrak{G}$ by *substituting* $\mathfrak{G}$ *by an isomorphic copy of itself.*

- **generalization and specialization**

  Let $\mathfrak{G} := (V, E, \nu, \top, Cut, area, \kappa, \rho)$ be a concept graph and $v_0 \in V$ be a vertex and $e_0 \in E$ an edge. Let $C \in \mathcal{C}$ be a concept name such that $C \geq \kappa(v)$ and $R \in \mathcal{R}$ be a relation name such that $R \geq \kappa(e)$. Set $\mathfrak{G}' := (V, E, \nu, \top, Cut, area, \kappa', \rho)$ with

$$\kappa'(k) := \begin{cases} \kappa(k) & k \neq v_0 \\ C & k = v_0 \end{cases} .$$

If $ctx(v_0)$ is a positive context, then we say that $\mathfrak{G}'$ is derived from $\mathfrak{G}$ by *generalizing the concept name of* v_0, and if $ctx(v_0)$ is a negative context, then we say that $\mathfrak{G}$ is derived from $\mathfrak{G}'$ by *specializing the concept name of* v_0.

Analogously for e, set $\mathfrak{G}'' := (V, E, \nu, \top, Cut, area, \kappa'', \rho)$ with

$$\kappa''(k) := \begin{cases} \kappa(k) & k \neq e_0 \\ R & k = e_0 \end{cases} .$$

If $ctx(e_0)$ is a positive context, then we say that $\mathfrak{G}'$ is derived from $\mathfrak{G}$ by *generalizing the relation name of* e_0. If $ctx(e_0)$ is a negative context, then we say that $\mathfrak{G}$ is derived from $\mathfrak{G}'$ by *specializing the relation name of* e_0.

- **exchanging references**

  Let $\mathfrak{G} := (V, E, \nu, \top, Cut, area, \kappa, \rho)$ be a concept graph and let $e \in E^{id}$ be an identity link with $ctx(e) = ctx(e\big|_1) = ctx(e\big|_2)$.

  Let $\mathfrak{G}' := (V, E, \nu, \top, Cut, area, \kappa, \rho')$ with

  $$\rho' : V \to V, \quad \rho'(v) := \begin{cases} \rho(v) & v \neq e\big|_1, e\big|_2 \\ \rho(e\big|_2) & v = e\big|_1 \\ \rho(e\big|_1) & v = e\big|_2 \end{cases} .$$

  Then we say that $\mathfrak{G}'$ is derived from $\mathfrak{G}$ by *exchanging the references of* v_1 *and* v_2.

- **merging two vertices and splitting a vertex**

  Let $\mathfrak{G} := (V, E, \nu, \top, Cut, area, \kappa, \rho)$ be a concept graph. Let v_1, v_2 be two vertices with $\rho(v_1) = \rho(v_2)$ $(\in \mathcal{G})$, $ctx(v_1) \geq ctx(v_2)$, and $\kappa(v_2) = \top$. Let $\mathfrak{G}' := (V', E', \nu', \top', Cut', area', \kappa', \rho')$ be the following graph:

  - $V' := V \backslash \{v_2\}$
  - $E' := E$
  - ν' is defined as follows: For $\nu(e)\big|_i = v$ let $\nu'(e)\big|_i = \begin{cases} v & v \neq v_2 \\ v_1 & v = v_2 \end{cases}$.
  - $Cut' := Cut$
  - $\top' := \top$
  - $area'(d) := area(d) \backslash \{v_2\}$ for all $d \in Cut' \cup \{\top'\}$
  - $\kappa' := \kappa|_{V' \cup E'}$
  - $\rho' := \rho|_{V'}$

  Then we say that $\mathfrak{G}'$ is derived from $\mathfrak{G}$ by *merging* v_2 *into* v_1 and $\mathfrak{G}$ is derived from $\mathfrak{G}'$ by *splitting* v_1.

Based on the calculus, we can now define the syntactical entailment relation.

Definition 5.2 (Syntactical Entailment Relation).

Let $\mathfrak{G}_a$*,* $\mathfrak{G}_b$ *be two nonexistential concept graphs. Then* $\mathfrak{G}_b$ can be derived from $\mathfrak{G}_a$ *(which is written* $\mathfrak{G}_a \vdash \mathfrak{G}_b$*), if there is a finite sequence* $(\mathfrak{G}_1, \mathfrak{G}_2, \ldots, \mathfrak{G}_n)$ *with* $\mathfrak{G}_1 = \mathfrak{G}_a$ *and* $\mathfrak{G}_b = \mathfrak{G}_n$ *such that each* $\mathfrak{G}_{i+1}$ *is derived from* $\mathfrak{G}_i$ *by applying one of the rules of the calculus. The sequence is called* a proof for $\mathfrak{G}_a \vdash \mathfrak{G}_b$*. Two graphs* $\mathfrak{G}_1, \mathfrak{G}_2$ *with* $\mathfrak{G}_1 \vdash \mathfrak{G}_2$ *and* $\mathfrak{G}_2 \vdash \mathfrak{G}_1$ *are said to be* provably equivalent*.*

If $\mathfrak{H} := \{\mathfrak{G}_i \,|\, i \in I\}$ *is a (possibly empty) set of nonexistential concept graphs, then* a graph $\mathfrak{G}$ can be derived from $\mathfrak{H}$ *if there is a finite subset* $\{\mathfrak{G}_1, \ldots, \mathfrak{G}_n\} \subseteq \mathfrak{H}$ *with* $\mathfrak{G}_1 \ldots \mathfrak{G}_n \vdash \mathfrak{G}$ *(remember that* $\mathfrak{G}_1 \ldots \mathfrak{G}_n$ *is the juxtaposition of* $\mathfrak{G}_1, \ldots, \mathfrak{G}_n$*).*

5.2 Remarks

In this section we provide an overview and some heuristics for the calculus we presented.

First we want to point out that all the rules are in some sense dually symmetric with respect to positive and negative contexts (this will be discussed in Detail in Sect. 14.4). More precisely, every rule which can be applied in one direction in positive contexts can be applied in the opposite direction in negative contexts, and vice versa. So if a rule can only be applied in positive contexts, this rule has a counterpart for negative contexts (see the pairs iteration/deiteration and specialization/generalization). All other rules apply to both positive and negative contexts.

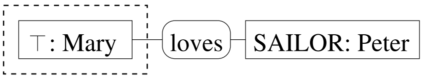
The first five rules (erasure, insertion, iteration, deiteration, double cut) are a ‘concept graph version’ of the original rules of Peirce’s calculus for existential graphs. The further rules are needed to encompass the orders on the concept- and relation names, to encompass the special properties of the concept name $\top$ and the relation name $\doteq$ and to deal with the possibility that different vertices can have the same reference.

In the following we go into detail for each rule. In particular we will provide some examples on how the rules are applied, and we will provide examples to show why some of the restrictions in the rules are necessary.

- **erasure, insertion**

  This is the first pair of rules which are dually symmetric to each other. Of course the operation ‘erasure’ is inverse to the operation ‘insertion’. But the *rule* ‘erasure’ may only be applied in *positive* contexts, and the *rule* ‘insertion’ may only be applied in *negative* contexts. Hence an application of the rule 'erasure' can *not* be reversed with an application of the rule 'insertion'. So if a concept graph $\mathfrak{G}'$ is derived from $\mathfrak{G}$ with one of the rules 'erasure' or 'insertion', then we usually have that $\mathfrak{G}'$ is more general than $\mathfrak{G}$ (i.e., $\mathfrak{G} \models \mathfrak{G}'$, but $\mathfrak{G}' \not\models \mathfrak{G}$). Therefore, these rules cause a loss of information and are called *generalization operations* or *generalization transformations*.

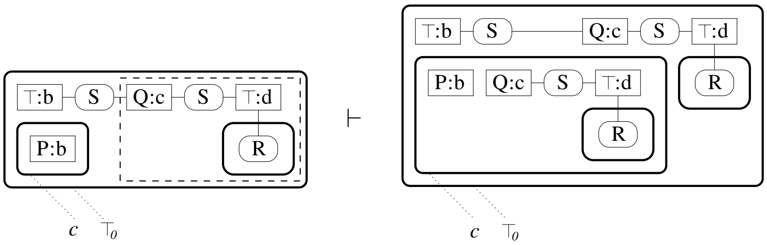
The restriction to *closed* subgraphs is necessary. To see this, have a look at the following simple graph with its marked subgraph:



When this subgraph is erased, we get a structure which is no concept graph at all.

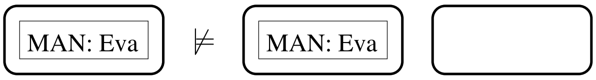
We want to emphasize that in Peirce's calculus, these two rules are the *only* generalization transformations, and in our calculus, the four rules 'erasure/insertion' and 'generalization/specialization' are the only generalization transformations. All other rules do not change the information a graph provides (the *conceptual content*, as Wille defines in [79] and [80]), i.e., they transform a graph into an equivalent graph (and are therefore called *equivalence operations* or *equivalence transformations*).

- **iteration, deiteration**
  - We start with an example for the iteration-rule. The subgraph which is marked by the dotted line is iterated into the cut c.



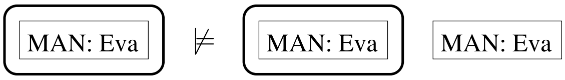
  Please note that c is a positive cut, hence we cannot derive the right from the left graph with an application of the insertion-rule.

  - If we iterate a cut, we have to iterate all what is scribed inside the cut, too.[1] To see this, consider the following example of an 'incomplete iteration':

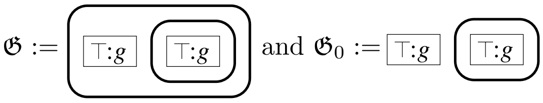


[1] This condition is captured by the definition of 'subgraph'.

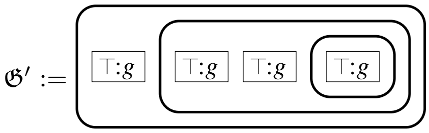
- To see that the condition $c \leq ctx(\mathfrak{G}_0)$ is necessary to keep the iteration-rule sound, consider the following example where we have iterated a concept box 'out of its cut':



- To see that the condition $c \notin Cut_0$ is necessary, consider the following graph $\mathfrak{G}$ with its subgraph $\mathfrak{G}_0$ (where g is an arbitrary object name):



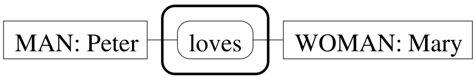
  If we iterate $\mathfrak{G}_0$ into the inner cut of $\mathfrak{G}$ (which belongs to $\mathfrak{G}_0$, hence the condition $c \notin Cut_0$ is violated), we get the following graph $\mathfrak{G}'$:



  It is easy to see that we $\mathfrak{G}$ is valid in every contextual model, but $\mathfrak{G}'$ is *not* valid in any contextual model, hence we have $\mathfrak{G} \not\models \mathfrak{G}'$.

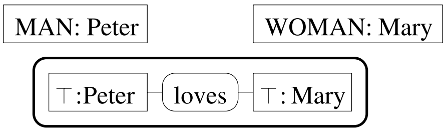
- The rules 'splitting a vertex' and 'merging two vertices' allow us to apply the iteration-rule to substructures which have 'pending edges' (i.e., edges such that some of the adjacent vertices do not belong to the substructure) and therefore are *no* subgraphs. We exemplify this with the following example:

  Consider the graph on the right.

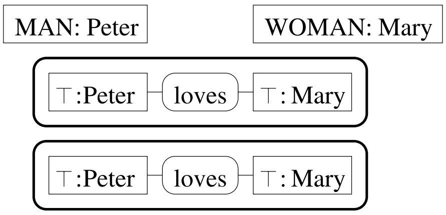


  We want to iterate the cut with its enclosed relation 'loves'. This is no subgraph.

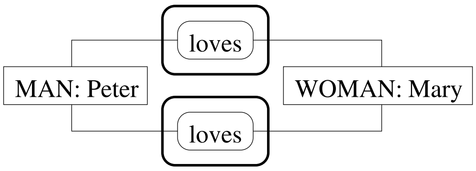
  First, we split the two vertices which cause the substructure to be no subgraph.



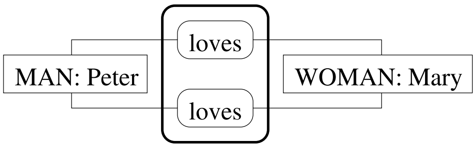
  Now we iterate the derived and *closed* subgraph.



After the iteration, the splitted vertices and their copies are merged (back) into their origins.



The whole proof can be carried out in both directions, i.e., the first graph can be derived from the last one. Moreover, a different application of the iteration-rule in step 3 yields that the graph we started with is provably equivalent to the following graph.



- Finally we want to point the following: When a subgraph is iterated, we generate for each vertex v of the subgraph a copy v'. As both v and v' carry the same object name, it it clear that they should refer to the same object. It will not be clear any longer, when in Chap. 10 of the part Beta the calculus for *existential* concept graphs with cuts is presented. In this calculus, it is allowed to draw an additional identity link between v and v'. This is in accordance to the iteration-rule in the part Beta of EGs, which is (in order to encompass the lines of identity) a refined version of iteration-rule of the part Alpha. In other words: The iteration- and deiteration-rules for nonexistential concept graphs correspond to the iteration- and deiteration-rules in the part Alpha of EGs, and the iteration- and deiteration-rules for existential concept graphs correspond to the iteration- and deiteration-rules in the part Beta of EGs.

– **double cut**

If we derive a concept graph from the empty sheet of assertion, we can only start with two rules: double cut and $\top$-insertion. The latter rule provides only few possibilities to proceed. Experience shows that most proofs start with adding a pair of cuts to the sheet of assertion with the double-cut-rule. After this step, an appropriate graph is added to the area of the outer cut with the insertion-rule. Lots of examples for proofs like this can be found in Sect. 10.3.

– **generalization and specialization, $\top$-erasure and $\top$-insertion**

All these rules encompass the orders on the concept and relation names.

In other works, the rules ‘generalization’ and ‘specialization’ are sometimes called ‘restriction’ and ‘unrestriction’, instead (see for example [62]).

The two rules ‘generalization’ and ‘specialization’ make up the other pair of rules which are generalization transformations and dually symmetric to each other (the first pair is ‘erasure’ and ‘insertion’). One of the rules can be omitted in the calculus (see Proposition 6.20), but we have added both rules in order to keep the calculus symmetric (see Chap. 14). However, it is important to note that we have an asymmetry in the alphabet $\mathcal{A}$: We can generalize each concept name to the name $\top$, but we do not have a concept name $\bot$.

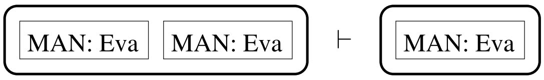
Moreover, we know more about $\top$ than that it is the greatest element of all concept-names. The symbol ‘$\top$’ stands for ‘true’, it is the ‘universal type’ (see [59]). This is reflected by the semantical interpretation of ‘$\top$’ in Def. 4.3: The $\vec{\mathbb{K}}$-interpretation $\lambda_{\mathcal{G}}$ maps $\top$ to the concept which has every object in its extension. Hence the symbol $\top$ has a special meaning (this has already been stressed by Wermelinger in [70]). As we have just elaborated, this special meaning is not covered by the rules ‘generalization’ and ‘specialization’ and must therefore be reflected by other rules in the calculus. For this reason, the $\top$-rules (i.e., $\top$-erasure and $\top$-insertion) are necessary. In contrast to ‘generalization’ and ‘specialization’, these rules are equivalence transformations.

- **isomorphism**

  The calculus we present here should be understood as a *diagrammatic calculus*, i.e., all rules can be carried out by manipulating the *diagrams* of concept graphs with cuts. But a diagram of a concept graph does not determine the mathematical objects which are represented by concept boxes or relation ovals, it only determines the relations between these objects. To see an example for this consideration, we consider the graph $\mathfrak{G}$ and its diagram which is shown in Fig. 2.1 on p. 33. In the definition of $\mathfrak{G}$ we did not specify the mathematical objects v_1, v_2, e_1, e_2, c_1, c_2 and $\top$ of the graph. In fact we can choose arbitrary sets for these mathematical objects (we only have to take into account that $\{v_1, v_2\}$, $\{e_1, e_2\}$ and $\{c_1, c_2, \top\}$ must be pairwise disjoint).

  In other words: The diagram of a graph determines the graph only up to isomorphism, and for this reason the isomorphism-rule is needed. We want to stress that this is the only rule of the calculus of non-diagrammatic character. As we will provide examples for concept graphs with cuts by their diagrams, and as we will carry out proofs with concept graphs in diagrammatic form, we have the implicit agreement that we usually regard concept graphs with cuts only up to isomorphism, i.e., concept graphs which are isomorphic are identified.

  This is particularly crucial when the rule ‘deiteration’ is applied to a graph. In order so see this, consider the following application of the deiteration-rule:



On the left, we have the diagram of the following graph:

$$\begin{aligned}\mathfrak{G}_l := (&\{v_1, v_2\}, \emptyset, \emptyset, \top_l, \{c\}, \{(\top_l, \{c\}), (c, \{v_1, v_2\}),\\ &\{(v_1, \mathrm{MAN}), (v_2, \mathrm{MAN})\},\ \{(v_1, \mathrm{Eva}), (v_2, \mathrm{Eva})\})\end{aligned}$$

The diagram does not specify the mathematical objects $v_1, v_2, c, \top_l$.

The graph of the diagram on the right is:

$$\mathfrak{G}_r := (\{v\}, \emptyset, \emptyset, \top, \{d\},\ \{(\top_r, \{d\}), (d, \{v\}), \{(v, \mathrm{MAN})\},\ \{(v, \mathrm{Eva})\})$$

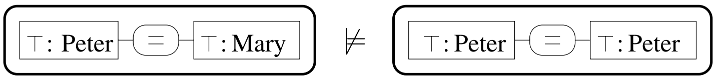
As we suppose that $\mathfrak{G}_r$ is derived from $\mathfrak{G}_l$ with the deiteration-rule, we know that $\mathfrak{G}_l$ is derived from $\mathfrak{G}_r$ with the iteration-rule (applied to the subgraph which contains only the vertex v). An application of the iteration-rule to $\mathfrak{G}_r$ yields in fact:

$$\begin{aligned}\mathfrak{G}_i := (&\{(v,1), (v,2)\}, \emptyset, \emptyset, \top_r, \{(d,1)\},\\ &\{(\top, \{(d,1)\}), ((d,1), \{(v,1), (v,2)\}),\\ &\{((v,1), \mathrm{MAN}), ((v,2), \mathrm{MAN})\},\ \{((v,1), \mathrm{Eva}), ((v,2), \mathrm{Eva})\})\end{aligned}$$

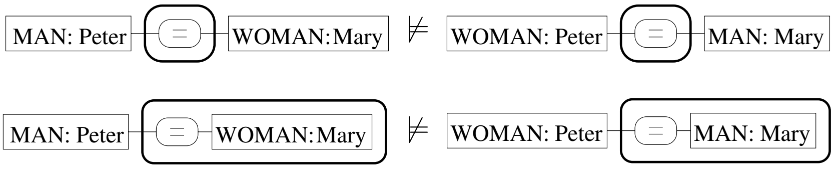
So we see the following: In order to apply the deitaration-rule to $\mathfrak{G}_l$, we must have objects v and d such that $v_1 = (v,1)$, $v_2 = (v,2)$ and $c = (d,1)$. This is not necessarily the case. Thus, before we can apply the deiteration-rule to $\mathfrak{G}_l$, we first had to apply the isomorphism-rule which transforms $\mathfrak{G}_l$ into the isomorphic graph $\mathfrak{G}_i$.

The best way to avoid this very technical problems is simply identifying isomorphic graphs. Hence, in the following, we will not mention anymore any application of the isomorphism-rule.

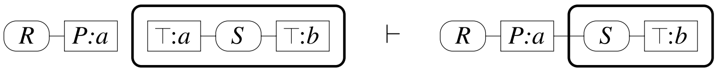
- **exchanging references**
  - This rule replaces the congruence-rule we presented in [13]. In this congruence rule, one reference is replaced by the other one (instead of exchanging the references). Unfortunately, a simple example shows that this congruence-rule is not sound:

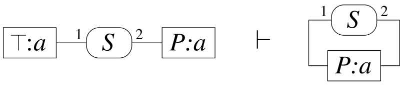


  - To see that the condition $ctx(e) = ctx(e\big|_1) = ctx(e\big|_2)$ is necessary, consider the following examples:

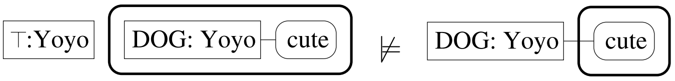


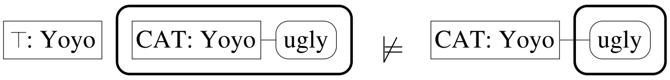
- **identity-erasure and identity-insertion**
  - We decided to replace the coreference-links of conceptual graphs by identity links in concept graphs. The reason for this can be found in the motivation for this treatise (Chap. 1) and in Chap. 14. A first, simple reason is that identity links can be treated like any other links. For instance, we can apply the rules ‘specialization’ and ‘generalization’ to identity-links (this would be more complicated if we used coreference-links). Nevertheless, the relation-name $\doteq$ (and therefore identity links) has a special meaning which is captured by its interpretation: In Def. 4.3 we assign the identity relation to the relation name $\doteq$. The special meaning of $\doteq$ must be reflected by rules in the calculus. This can be seen in the rules ‘exchanging references’, ‘splitting a vertex’ and ‘merging two vertices’, where identity links play a special role. However, with these rules it is not possible to add or remove redundant identity links. Therefore the rules ‘identity-erasure’ and ‘identity-insertion’ are needed as well.
  - Please note that we did not specify how an identity link e which is added between two vertices v_1, v_2 with the identity-insertion-rule is directed (i.e., whether we have $e|_1 = v_1$, $e|_2 = v_2$ or $e|_1 = v_2$, $e|_2 = v_1$). Of course this is based on the symmetry of identity.
  - The condition $c < ctx(v_1), ctx(v_2)$ in the rule ‘identity-insertion’ is only needed in order to obtain a concept graph with dominating nodes.
- **merging two vertices and splitting a vertex**
  - We start with two examples for these rules. In both examples, the right graph is derived from the left graph by merging $\boxed{P : a}$ into $\boxed{\top : a}$. Hence in both examples the right graph can be derived from the left graph by splitting the vertex $\boxed{P : a}$. In the right graph of the second example, we have only one vertex $\boxed{P : a}$, but it occurs *twice* on the edge. So, in order to derive the left graph from the right one, only *one* of these two occurences is substituted by $\boxed{\top : a}$.





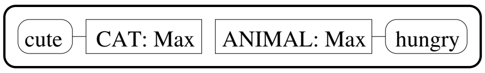
  - In order to see that the condition $ctx(v_1) \geq ctx(v_2)$ in the rule ‘merging two vertices’ is necessary, consider the following two simple examples (we assume that Yoyo is a cute cat):



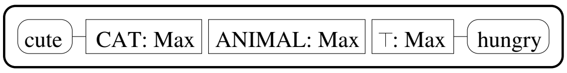


- The rules 'merging two vertices' and 'splitting a vertex' deal with the fact that one object may be the reference for different vertices. With these rules it is possible to transform every concept graph into an equivalent graph in which no cut intersects a relation line (see Lem. 5.8).
- It is easy to see that a vertex $v_1 = \boxed{P_1 : g}$ may be merged into a vertex $v_2 = \boxed{P_2 : g}$, if both vertices are placed in the same *positive* context (this can be done as follows: First v_1 is generalized to $v_1' = \boxed{\top : g}$ and then v_1' is merged into v_2). As the generalization rule may only applied in positive contexts, this is not allowed in negative contexts. However, the following may be done: If $v_1 = \boxed{P_1 : g}$ and $v_2 = \boxed{P_2 : g}$ are placed in the same negative context and if $P_1 \leq P_2$ holds, then v_2 may be merged into v_1. We exemplify this with an example:

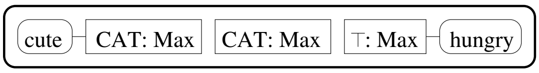
  In the graph on the right, we want to merge right vertex into the left one.

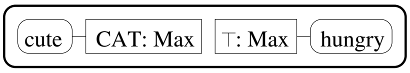
  We split the vertex $\boxed{\text{ANIMAL: Max}}$.

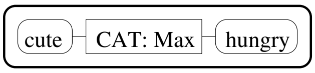
  The concept name ANIMAL is specialized to CAT.

  The isolated vertex $\boxed{\text{CAT: Max}}$ is now erased by using the deiteration-rule.

  We merge $\boxed{\top\text{: Max}}$ into $\boxed{\text{CAT: Max}}$.

  So, finally we have merged $\boxed{\text{ANIMAL: Max}}$ into $\boxed{\text{CAT: Max}}$.
- We want to point out that the rules 'merging two vertices' and 'splitting a vertex' are often used in a specific manner. First, the rule 'splitting a vertex' is applied to transform a graph into an equivalent graph such that a certain desired rule can be applied to the new graph. Then this desired rule is carried out. Finally the application of the rule 'splitting a vertex' is reversed with the rule 'merging two vertices'.

We have already provided an example for this strategy in our remarks for the iteration- and deiteration-rule. In this example, we wanted to iterate a substructure which was no subgraph. So we used the rule 'splitting a vertex' to transform the substructure into a (closed) subgraph. Then we were allowed to iterate this closed subgraph. Afterwards we used the rule 'merging two vertices' to merge the splitted vertices and their copies (the copies we obtained from the iteration-rule) back into their origins.

5.3 Theorems and Normal Forms

In [42] Peirce provided 16 useful transformation rules for existential graphs which he derived from his calculus. These rules are logical metalemmata in the sense that they show some *schemata* for proofs with existential graphs, i.e., they are derived 'macro'-rules. In this section we provide the concept graph versions for two of these transformation rules. We start with a (weakened) version of the first transformation rule of Peirce.

Lemma 5.3 (Reversion Theorem).

Let $\mathfrak{G}_a$ and $\mathfrak{G}_b$ be two nonexistential concept graphs. Then we have $\mathfrak{G}_a \vdash \mathfrak{G}_b$ if and only if $\boxed{\mathfrak{G}_b} \vdash \boxed{\mathfrak{G}_a}$.

Proof: Let $(\mathfrak{G}_1, \mathfrak{G}_2, \ldots, \mathfrak{G}_n)$ with $\mathfrak{G}_1 = \mathfrak{G}_a$ and $\mathfrak{G}_b = \mathfrak{G}_n$ be a proof for $\mathfrak{G}_a \vdash \mathfrak{G}_b$. Then, due to the symmetry of the calculus, ($\boxed{\mathfrak{G}_n}$, $\boxed{\mathfrak{G}_{n-1}}$, …, $\boxed{\mathfrak{G}_1}$) is a proof for $\boxed{\mathfrak{G}_b} \vdash \boxed{\mathfrak{G}_a}$. The inverse direction holds as well. □

All rules in the calculus which are applied in a context only depend on whether the context is positive or negative. In particular if a proof for $\mathfrak{G}_a \vdash \mathfrak{G}_b$ is given, this proof can be carried out in arbitrary positive contexts. Together with the previous lemma, this yields the following lemma, which is a combination of the (full) first and the sixth transformation rule of Peirce. It can also be found in [62] (from where we adopted the name of the theorem).

Lemma 5.4 (Cut-And-Paste-Theorem).

Let $\mathfrak{G}$ be a valid nonexistential concept graph and let $\mathfrak{G}_a, \mathfrak{G}_b$ be nonexistential concept graphs. Then we have:

- *If $\mathfrak{G}_a$ is a closed subgraph of the graph $\mathfrak{G}$ in a positive context, then the subgraph $\mathfrak{G}_a$ of $\mathfrak{G}$ may be replaced by $\mathfrak{G}_b$, and the resulting graph is still valid.*
- *If $\mathfrak{G}_b$ is a closed subgraph of the graph $\mathfrak{G}$ in a negative context, then the subgraph $\mathfrak{G}_b$ of $\mathfrak{G}$ may be replaced by $\mathfrak{G}_a$, and the resulting graph is still valid.*

In particular we have that derivable graphs $\mathfrak{G}_0$ (i.e., graphs with $\vdash \mathfrak{G}_0$) can be inserted into arbitrary contexts of arbitrary graphs.

Without Proof.

Neither in nonexistential concept graphs with cuts, nor in existential concept graphs with cuts, we have free vraiables. So it has to be expected that we can show a concept graph version of the well known deduction theorem. This is done is the next lemma.

Lemma 5.5 (Deduction Theorem).

Let $\mathfrak{G}_a$, $\mathfrak{G}_b$ be two concept graphs. Then $\mathfrak{G}_a \vdash \mathfrak{G}_b \iff \vdash \boxed{\mathfrak{G}_a \boxed{\mathfrak{G}_b}}$

Proof: We show both directions separately.

'⟹':

$$\overset{\text{double cut}}{\vdash} \boxed{\boxed{\phantom{\mathfrak{G}_b}}}$$

$$\overset{\text{insertion}}{\vdash} \boxed{\mathfrak{G}_a \boxed{\phantom{\mathfrak{G}_b}}}$$

$$\overset{\text{iteration}}{\vdash} \boxed{\mathfrak{G}_a \boxed{\mathfrak{G}_a}}$$

$$\overset{\text{Lem. 5.4}}{\vdash} \boxed{\mathfrak{G}_a \boxed{\mathfrak{G}_b}}$$

'⟸':

$$\mathfrak{G}_a \overset{\text{Lem. 5.4}}{\vdash} \mathfrak{G}_a \boxed{\mathfrak{G}_a \boxed{\mathfrak{G}_b}}$$

$$\overset{\text{deiteration}}{\vdash} \mathfrak{G}_a \boxed{\phantom{\mathfrak{G}_a} \boxed{\mathfrak{G}_b}}$$

$$\overset{\text{double cut}}{\vdash} \mathfrak{G}_a \quad \mathfrak{G}_b$$

$$\overset{\text{erasure}}{\vdash} \mathfrak{G}_b$$

□

The following lemma is quite obvious:

Lemma 5.6 (Derivation of Juxtapositions).

Let $\mathfrak{G}$, $\mathfrak{G}_a$ and $\mathfrak{G}_b$ be three concept graphs with $\mathfrak{G} \vdash \mathfrak{G}_a$ and $\mathfrak{G} \vdash \mathfrak{G}_b$. Then $\mathfrak{G} \quad \vdash \quad \mathfrak{G}_a \ \mathfrak{G}_b$.

Proof: $\mathfrak{G} \overset{\text{iteration}}{\vdash} \mathfrak{G}\ \mathfrak{G} \overset{\text{Lem. 5.4}}{\vdash} \mathfrak{G}_a\ \mathfrak{G} \overset{\text{Lem. 5.4}}{\vdash} \mathfrak{G}_a\ \mathfrak{G}_b$ □

We want to stress that in the preceding lemmata, only the first five rules of the calculus (i.e., erasure, insertion, iteration, deiteration, double cut) have been used. This will be important in Sect. 6.3, where we will use these lemmata in some restricted calculi.

A well-known standard form for a concept graph is its *normal form* which requires that each object name appears at most once as a reference. We will introduce another standard form for concept graphs which is quite contrary to the normal-form (it is therefore called *anti-normal-form*). These notions are captured by the following definition.

Definition 5.7 (Normal-form and Anti-Normal-Form).

A concept graph is said to be in normal-form *if it satisfies the following condition:* $\rho(v_1) = \rho(v_2) \in \mathcal{G} \Longrightarrow v_1 = v_2$.

A concept graph is called free of intersections*, if it satisfies the following condition:* $\forall e \in E . \forall v \in V . v \in V_e \Longrightarrow ctx(v) = ctx(e)$.

A concept graph which is free of intersections is said to be in anti-normal-form*, if it satisfies the following conditions:*

- $\forall e \in E . \forall v \in V . v \in V_e \Longrightarrow \kappa(v) = \top$
- $\forall e, f \in E . \forall i, j \in \mathbb{N} . e \neq f \vee i \neq j \Longrightarrow e_i \neq f_j$

Graphs in anti-normal-form can be understood as fragmented into elementary graphs (i.e., graphs which carry atomic information) and are therefore easy to read. The next lemma shows that, using the rules 'merging two vertices' and 'splitting a vertex', each concept graph can be transformed into an equivalent graph in anti-normal-form.

Lemma 5.8 (Transformation of Graphs into Anti-Normal-Form).

Each concept graph $\mathfrak{G}$ is provably equivalent to a concept graph in anti-normal-form.

Proof: Let $\mathfrak{G} := (V, E, \nu, \top, Cut, area, \kappa, \rho)$. For each edge $e = (v_1, \ldots, v_n)$, $e \in E$ and every occurence[2] of a vertex $v_i \in V_e$, a new vertex $v_i' = \boxed{\top : \rho(v_i)}$ is inserted into $ctx(e)$ and $e|_i = v_i$ is replaced by $e|_i := v_i'$ (this is an application of the rule 'splitting a vertex'). Because all these transformations may be reversed (with the rule 'merging two vertices') the new constructed graph is equivalent to $\mathfrak{G}$. Obviously, the new graph is in anti-normal-form. □

Example 5.9. The following four graphs are equivalent. The first graph is in normal-form, but is it not free of intersections. The second graph is free of intersections, but neither in normal nor in anti-normal-form. The third graph is neither free of intersections, nor in anti-normal-form, nor in normal-form. The fourth graph is in anti-normal-form.

[2] In order to see that for every *occurence* a new concept box $\boxed{\top : \rho(v_i)}$ has to be inserted, look at the second example for the rules 'merging two vertices' and 'splitting a vertex' on p. 57. In this example, we must insert *two* new vertices.

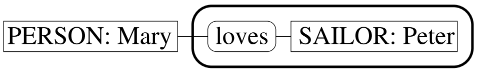


Fig. 5.1. A graph which is in normal-form, but which ist not free of intersections.

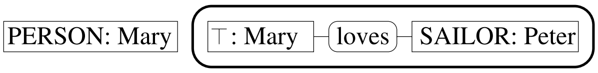


Fig. 5.2. A graph which is free of intersections, but neither in normal nor in anti-normal-form.

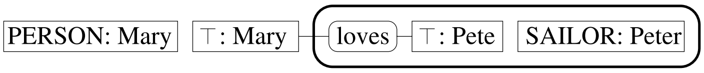


Fig. 5.3. A graph which is neither free of intersections, nor in anti-normal-form, nor in normal-form.

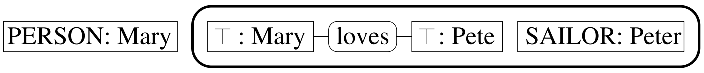


Fig. 5.4. A graph which is in anti-normal-form.

6 Soundness and Completeness

In this chapter we will show that the rules of Chap. 5 are sound and complete with respect to the given semantics. In the first section, we will prove the soundness of the calculus, in the next section we will prove its completeness. In the last section we will show that all rules we have added to the calculus of Peirce are needed. More precisely: If a correlated pair of rules (i.e., generalization and specialization, $\top$-erasure and $\top$-insertion, identity-erasure and identity-insertion, merging two vertices and splitting a vertex) or the rule 'exchanging references' is removed, then the calculus is not complete anymore.

6.1 Soundness

Most of the rules modify only the area of one specific context c (for example, the rule 'erasure' removes edges or subgraphs from the area of a positive context). If a graph $\mathfrak{G}'$ is derived from a graph $\mathfrak{G}$ by applying one of these rules (i.e., by modifying a context c in the graph $\mathfrak{G}$), $\mathfrak{G}$ and $\mathfrak{G}'$ are isomorphic except for c. As it has to be shown that no rule can transform a valid graph into a nonvalid one, the next theorem is the basis for proving the soundness of most rules.

Theorem 6.1 (Main Theorem for Soundness of Alpha).
Let $\mathfrak{G} := (V, E, \nu, \top, Cut, area, \kappa, \rho)$, $\mathfrak{G}' := (V', E', \nu', \top', Cut', area', \kappa', \rho')$ be two nonexistential concept graphs with cuts and let $f = f_V \dot\cup f_E \dot\cup f_{Cut}$ be an isomorphism from $\mathfrak{G}$ to $\mathfrak{G}'$ except for $c \in Cut \cup \{\top\}$ and $c' \in Cut' \cup \{\top'\}$. Let $(\vec{\mathbb{K}}, \lambda)$ be a contextual model. Let $P(d)$ be the following property for Cuts $d \in Cut \cup \{\top\}$:

- *If d is positive and $(\vec{\mathbb{K}}, \lambda) \models \mathfrak{G}[d]$, then $(\vec{\mathbb{K}}, \lambda) \models \mathfrak{G}'[f(d)]$, and*
- *If d is negative and $(\vec{\mathbb{K}}, \lambda) \not\models \mathfrak{G}[d]$, then $(\vec{\mathbb{K}}, \lambda) \not\models \mathfrak{G}'[f(d)]$.*

If P holds for c, then P holds for each $d \in (Cut \cup \{\top\})$ with $d \not< c$. In particular follows $(\vec{\mathbb{K}}, \lambda) \models \mathfrak{G}'$ from $(\vec{\mathbb{K}}, \lambda) \models \mathfrak{G}$.

F. Dau: The Logic System of Concept Graphs with Negation, LNAI 2892, pp. 63-80, 2003.

Proof: We set $D := \{d \in Cut \cup \{\top\} \mid d \not< c\}$. D is a tree such that for each $d \in D$ with $d \neq c$ and each $e \in Cut \cup \{\top\}$ with $e < d$ we have $e \in D$. For this reason we can carry out the proof by induction over D. So let $d \in D$, $d \neq c$ (we know that c satisfies P) be a context such that $P(e)$ holds for all cuts $e \in area(d) \cap Cut$.

First Case: d is positive and $(\vec{\mathbb{K}}, \lambda) \models \mathfrak{G}[d]$.

We have to check the vertex-, edge- and cut-conditions for $f(d)$. We start with the vertex conditions for $f(d)$, i.e., for vertices $v' \in V'$ with $ctx'(v') = f(d)$.

For each $v \in V$ with $ctx(v) = d$, it holds $\rho(v) = \rho'(f(v))$ and $\kappa(v) = \kappa'(f(v))$, hence $\lambda_{\mathcal{G}}(\rho(v)) \in Ext(\lambda_{\mathcal{C}}(\kappa(v))) \Longleftrightarrow \lambda_{\mathcal{G}}(\rho'(f(v))) \in Ext(\lambda_{\mathcal{C}}(\kappa'(f(v))))$. As f_V is a bijection from $area(d) \cap V$ to $area'(f(d)) \cap V'$, we gain the following: All vertex conditions in d hold iff all vertex conditions in $f(d)$ hold.

The edge conditions are checked analogously.

From $(\vec{\mathbb{K}}, \lambda) \models \mathfrak{G}[d]$ we get $(\vec{\mathbb{K}}, \lambda) \not\models \mathfrak{G}[e]$ for all cuts $e \in area(d)$. These cuts are negative and are mapped bijectively to the cuts $e' \in area(f(d))$. As they are negative, we conclude from the induction hypothesis or the presupposition that $(\vec{\mathbb{K}}, \lambda) \not\models \mathfrak{G}'[f(e)]$ for all cuts $e \in area(d)$, i.e., $(\vec{\mathbb{K}}, \lambda) \not\models \mathfrak{G}'[e']$ for all cuts $e' \in area'(f(d))$.

As we have checked all conditions for $f(d)$, we get $(\vec{\mathbb{K}}, \lambda) \models \mathfrak{G}'[f(d)]$.

Second Case: d is negative and $(\vec{\mathbb{K}}, \lambda) \not\models \mathfrak{G}[d]$.

This is shown analogously to the first case. □

With this lemma, we can prove the correctness of the rules.

We start with the soundness of the rules 'iteration' and 'deiteration'.

Lemma 6.2 (Iteration and Deiteration are Sound).

If $\mathfrak{G}$ and $\mathfrak{G}'$ are two nonexistential concept graphs with cuts, $(\vec{\mathbb{K}}, \lambda)$ is a contextual structure with $(\vec{\mathbb{K}}, \lambda) \models \mathfrak{G}$ and $\mathfrak{G}'$ is derived from $\mathfrak{G}$ by applying one of the rules 'iteration' or 'deiteration', then $(\vec{\mathbb{K}}, \lambda) \models \mathfrak{G}'$.

Proof: Let $\mathfrak{G}_0 := (V_0, E_0, \nu_0, \top_0, Cut_0, area_0, \kappa_0, \rho_0)$ be the subgraph of $\mathfrak{G}$ which is iterated into the context $c \leq ctx(\mathfrak{G}_0)$, $c \notin Cut_0$. We use the mathematical notation which was given in Sect. 5.1. In particular, $(c, 1)$ is the context in $\mathfrak{G}'$ which corresponds to the context c in $\mathfrak{G}$. There are two cases to consider:

First Case: $(\vec{\mathbb{K}}, \lambda) \models \mathfrak{G}_0$. From this we conclude the following equivalence: $(\vec{\mathbb{K}}, \lambda) \models \mathfrak{G}[c] \Longleftrightarrow (\vec{\mathbb{K}}, \lambda) \models \mathfrak{G}'[(c, 1)]$. As $\mathfrak{G}$ and $\mathfrak{G}'$ are isomorphic except for the contexts $c \in Cut \cup \{\top\}$ and $(c, 1) \in Cut' \cup \{\top'\}$, Thm. 6.1 can be applied now. This yields

$$(\vec{\mathbb{K}}, \lambda) \models \mathfrak{G} \Longleftrightarrow (\vec{\mathbb{K}}, \lambda) \models \mathfrak{G}' \ . \qquad (*)$$

Second Case: $(\vec{\mathbb{K}},\lambda) \not\models \mathfrak{G}_0$. From this we conclude $(\vec{\mathbb{K}},\lambda) \not\models \mathfrak{G}[\top_0]$ and $(\vec{\mathbb{K}},\lambda) \not\models \mathfrak{G}'[(\top_0,1)]$. As $\mathfrak{G}$ and $\mathfrak{G}'$ are isomorphic except for the contexts $\top_0 \in Cut$ and $(\top_0,1) \in Cut'$, Thm. 6.1 can be applied now, which yields $(*)$.

The direction '$\Longrightarrow$' of $(*)$ yields the correctness of the iteration-rule, the opposite direction '$\Longleftarrow$' yields the correctness of the deiteration-rule. □

Lemma 6.3 (Erasure and Insertion are Sound).

If $\mathfrak{G}$ and $\mathfrak{G}'$ are two nonexistential concept graphs with cuts, $(\vec{\mathbb{K}},\lambda)$ is a contextual structure with $(\vec{\mathbb{K}},\lambda) \models \mathfrak{G}$ and $\mathfrak{G}'$ is derived from $\mathfrak{G}$ by applying one of the rules 'erasure' or 'insertion', then $(\vec{\mathbb{K}},\lambda) \models \mathfrak{G}'$.

Proof: We start with 'erasure'. Let $\mathfrak{G}_0 := (V_0, E_0, \nu_0, \top_0, Cut_0, area_0, \kappa_0, \rho_0)$ be the subgraph which is erased (the soundness of erasing edges can be shown analogously). $\mathfrak{G}_0$ is erased from the area of the positive context $c := \top_0$. Obviously, if $(\vec{\mathbb{K}},\lambda) \models \mathfrak{G}[c]$, then $(\vec{\mathbb{K}},\lambda) \models \mathfrak{G}'[c]$. Furthermore $\mathfrak{G}$ and $\mathfrak{G}'$ are isomorphic except for $c \in Cut \cup \{\top\}$ and $c \in Cut' \cup \{\top'\}$, hence Thm. 6.1 can be applied now. This yields $(\vec{\mathbb{K}},\lambda) \models \mathfrak{G}'$.

The soundness of the insertion-rule is proven analogously. □

Lemma 6.4 (Double Cut is Sound).

If $\mathfrak{G}$ and $\mathfrak{G}'$ are two nonexistential concept graphs with cuts, $(\vec{\mathbb{K}},\lambda)$ is a contextual structure with $(\vec{\mathbb{K}},\lambda) \models \mathfrak{G}$ and $\mathfrak{G}'$ is derived from $\mathfrak{G}$ by applying the rule 'double cut', then $(\vec{\mathbb{K}},\lambda) \models \mathfrak{G}'$.

Proof: Let $\mathfrak{G}$ and $\mathfrak{G}'$ be two concept graphs such that $\mathfrak{G}$ is derived from $\mathfrak{G}'$ by erasing two cuts c_1, c_2 with $area(c_1) = \{c_2\}$. We set $c := ctx(c_1)$. We want to apply Thm. 6.1 and therefore have to show that property P of Thm. 6.1 is valid for $c \in \mathfrak{G}$ and $c \in \mathfrak{G}'$. We have

$$area'(c) = (area(c) \cup area(c_2)) \backslash \{c_1\} \qquad (*)$$

With $(*)$ we get

$(\vec{\mathbb{K}},\lambda) \models \mathfrak{G}[c]$

$\overset{\text{Def}\models}{\Longleftrightarrow}$ $(\vec{\mathbb{K}},\lambda)$ fulfills all vertex-, edge- and cut-conditions of $area(c)$

$\Longleftrightarrow$ $(\vec{\mathbb{K}},\lambda)$ fulfills all vertex-, edge- and cut-conditions of $area(c)\backslash\{c_1\}$ and $(\vec{\mathbb{K}},\lambda) \not\models \mathfrak{G}[c_1]$

$\Longleftrightarrow$ $(\vec{\mathbb{K}},\lambda)$ fulfills all vertex-, edge- and cut-conditions of $area(c)\backslash\{c_1\}$ and $(\vec{\mathbb{K}},\lambda) \models \mathfrak{G}[c_2]$

$\overset{(*)}{\Longleftrightarrow}$ $(\vec{\mathbb{K}},\lambda)$ fulfills all vertex-, edge- and cut-conditions of $area'(c)$

$\overset{\text{Def}\models}{\Longleftrightarrow}$ $(\vec{\mathbb{K}},\lambda) \models \mathfrak{G}'[c]$

Now Thm. 6.1 yields that we have $(\vec{\mathbb{K}},\lambda) \models \mathfrak{G} \Longleftrightarrow (\vec{\mathbb{K}},\lambda) \models \mathfrak{G}'$. □

Lemma 6.5 (Specialization and Generalization are Sound).

If $\mathfrak{G}$ and $\mathfrak{G}'$ are two nonexistential concept graphs with cuts, $(\vec{\mathbb{K}}, \lambda)$ is a contextual structure with $(\vec{\mathbb{K}}, \lambda) \models \mathfrak{G}$ and $\mathfrak{G}'$ is derived from $\mathfrak{G}$ by applying one of the rules 'specialization' or 'generalization', then $(\vec{\mathbb{K}}, \lambda) \models \mathfrak{G}'$.

Proof: Let $\mathfrak{G}'$ is derived from $\mathfrak{G}$ by generalizing the concept name of a vertex v_0. Let $c := ctx(v_0)$. Now, analogously to the proof of Lem. 6.3, we see that $\mathfrak{G}$ and $\mathfrak{G}'$ are isomorphic except for $c \in Cut \cup \{\top\}$ and $c \in Cut' \cup \{\top'\}$, and if $(\vec{\mathbb{K}}, \lambda) \models \mathfrak{G}[c]$, then $(\vec{\mathbb{K}}, \lambda) \models \mathfrak{G}'[c]$. Hence Thm. 6.1 yields $(\vec{\mathbb{K}}, \lambda) \models \mathfrak{G}'$.

The soundness of the specialization-rule is proven analogously. □

The remaining rules 'merging two vertices', 'splitting a vertex', 'identity-erasure', 'identity-insertion', '⊤-erasure', and '⊤-insertion', and 'exchanging references' do not change the 'informational content' of any cut at all. Thus, their soundness could be shown without an application of Thm. 6.1.

Lemma 6.6 (Merging and Splitting Vertices is Sound).

If $\mathfrak{G}$ and $\mathfrak{G}'$ are two nonexistential concept graphs with cuts, $(\vec{\mathbb{K}}, \lambda)$ is a contextual structure with $(\vec{\mathbb{K}}, \lambda) \models \mathfrak{G}$ and $\mathfrak{G}'$ is derived from $\mathfrak{G}$ by applying one of the rules 'merging two vertices' or 'splitting a vertex', then $(\vec{\mathbb{K}}, \lambda) \models \mathfrak{G}'$.

Proof: Again we use the mathematical notation which was given in Sect. 5.1. So let the concept graph $\mathfrak{G}' := (V', E', \nu', \top', Cut', area', \kappa', \rho')$ be derived from $\mathfrak{G} := (V, E, \nu, \top, Cut, area, \kappa, \rho)$ by merging v_2 into v_1.

Let $e \in E$ be an edge with $\nu(e)|_i = v$. Then we have $\nu'(e)\big|_i = \begin{cases} v & v \neq v_2 \\ v_1 & v = v_2 \end{cases}$.

As we have $\rho(v_2) = \rho(v_1)$, we get $\rho(e) = \rho'(e)$. So the edge conditions in $\mathfrak{G}$ and $\mathfrak{G}'$ are exactly the same.

The vertex conditions in $\mathfrak{G}$ and $\mathfrak{G}'$ are exactly the same, too, except for the vertex v_2 which is erased from $\mathfrak{G}$. As of $\kappa(v_2) = \top$, the vertex condition for v_2 in $\mathfrak{G}$ is trivially fulfilled (hence $(\vec{\mathbb{K}}, \lambda) \models \mathfrak{G}[c] \iff (\vec{\mathbb{K}}, \lambda) \models \mathfrak{G}'[c]$ for $c := ctx(v_2)$).

To summarize: In $\mathfrak{G}$ and $\mathfrak{G}'$, except for one trivially fulfilled vertex condition in $\mathfrak{G}$, the same vertex- and edge-conditions have to be checked. So, by induction over $Cut \cup \{\top\}$, we see that we have

$$(\vec{\mathbb{K}}, \lambda) \models \mathfrak{G}[d] \iff (\vec{\mathbb{K}}, \lambda) \models \mathfrak{G}'[d]$$

for each cut $d \in Cut$. In particular we have

$$(\vec{\mathbb{K}}, \lambda) \models \mathfrak{G} \iff (\vec{\mathbb{K}}, \lambda) \models \mathfrak{G}' \qquad (*)$$

The direction '$\Longrightarrow$' of $(*)$ yields the correctness of 'merging two vertices'-rule. The opposite direction '$\Longleftarrow$' of $(*)$ yields the correctness of 'splitting a vertex'-rule. □

Lemma 6.7 (Identity/$\top$-Erasure/Insertion are Sound).

If $\mathfrak{G}$ and $\mathfrak{G}'$ are two nonexistential concept graphs with cuts, $(\vec{\mathbb{K}}, \lambda)$ is a contextual structure with $(\vec{\mathbb{K}}, \lambda) \models \mathfrak{G}$ and $\mathfrak{G}'$ is derived from $\mathfrak{G}$ by applying one of the rules 'identity-erasure', 'identity-insertion', '$\top$-erasure', or '$\top$-insertion', then $(\vec{\mathbb{K}}, \lambda) \models \mathfrak{G}'$.

Proof: The identity condition for an identity link $id \in E^{id}$ with $\rho(id|_1) = \rho(id|_2)$ is trivially fulfilled. So we have a situation which is similar to the situation in Lem. 6.6, where an vertex, whose vertex-condition is trivially fulfilled, is inserted or erased. The situation is even simpler, as $ctx(e)$ is the only context which is changed. Hence the correctness of the rules 'identity-erasure' or 'identity-insertion' can be shown analogously to Lem. 6.6.

The same holds for the rules '$\top$-erasure' and '$\top$-insertion'. □

Lemma 6.8 (Exchanging References is Sound).

If $\mathfrak{G}$ and $\mathfrak{G}'$ are two nonexistential concept graphs with cuts, $(\vec{\mathbb{K}}, \lambda)$ is a contextual structure with $(\vec{\mathbb{K}}, \lambda) \models \mathfrak{G}$ and $\mathfrak{G}'$ is derived from $\mathfrak{G}$ by applying the rule 'exchanging references', then $(\vec{\mathbb{K}}, \lambda) \models \mathfrak{G}'$.

Proof: Let $e = (v_1, v_2) \in E^{id}$ be an identity link such that e and its incident vertices are placed in the same context c. Set $k_1 := \lambda_{\mathcal{G}}(\rho(v_1))$ and $k_2 := \lambda_{\mathcal{G}}(\rho(v_2))$. The identity link and its incident vertices lead to the following identity- and vertex conditions:

$$\text{in } \mathfrak{G} : \quad k_1 \in Ext(\lambda_{\mathcal{C}}(\kappa(v_1)))\,,\, k_2 \in Ext(\lambda_{\mathcal{C}}(\kappa(v_2))) \text{ and } k_1 = k_2$$

$$\text{in } \mathfrak{G}' : \quad k_2 \in Ext(\lambda_{\mathcal{C}}(\kappa(v_1)))\,,\, k_1 \in Ext(\lambda_{\mathcal{C}}(\kappa(v_2))) \text{ and } k_1 = k_2$$

These conditions are equivalent. We have that $\mathfrak{G}$ and $\mathfrak{G}'$ are isomorphic except for the context c, and, for each contextual structure $(\vec{\mathbb{K}}, \lambda)$, we have the equivalence $(\vec{\mathbb{K}}, \lambda) \models \mathfrak{G}[c] \Longleftrightarrow (\vec{\mathbb{K}}, \lambda) \models \mathfrak{G}'[c]$.

Again, an argumentation similar to the proof of 6.6 yields that the we have

$$(\vec{\mathbb{K}}, \lambda) \models \mathfrak{G} \Longleftrightarrow (\vec{\mathbb{K}}, \lambda) \models \mathfrak{G}' \ ,$$

i.e., the rule 'exchanging references' is sound. □

From the preceding lemmata the soundness of the calculus follows immediately:

Theorem 6.9 (Soundness of the Alpha-Calculus).

Two nonexistential, simple concept graph $\mathfrak{G}$, $\mathfrak{G}'$ with cuts over $\mathcal{A}$ satisfy

$$\mathfrak{G} \vdash \mathfrak{G}' \quad \Longrightarrow \quad \mathfrak{G} \models \mathfrak{G}'$$

6.2 Completeness

As the empty sheet of assertion is always true, the concept graph $\boxed{\quad}$ is always false.[1] This leads to the following definition:

Definition 6.10 (Consistent Sets of Graphs).

A set $\mathfrak{H}$ of concept graphs is called consistent *if $\mathfrak{H} \vdash \boxed{\quad}$ does not hold. A concept graph $\mathfrak{G}$ is called* consistent *if $\{\mathfrak{G}\}$ is consistent.*

The following lemma is well known:

Lemma 6.11.

A set $\mathfrak{H}$ of concept graphs $\mathfrak{G}$ is not consistent if and only if $\mathfrak{H} \vdash \mathfrak{G}'$ for every concept graph $\mathfrak{G}'$.

Proof: Only the direction '$\Longrightarrow$' has to be proven. So let $\mathfrak{G}_1, \ldots, \mathfrak{G}_n \in \mathfrak{H}$ with $\mathfrak{G}_1 \ldots \mathfrak{G}_n \vdash \boxed{\quad}$. Let $\mathfrak{G}$ be the juxtaposition of $\mathfrak{G}_1, \ldots, \mathfrak{G}_n$. We conclude:

$$\begin{array}{rcl}
\mathfrak{G} \vdash \boxed{\quad} & \overset{\text{Ded.Theorem}}{\Longleftrightarrow} & \vdash \boxed{\mathfrak{G} \ \boxed{\boxed{\quad}}} \\
 & \overset{\text{double cut}}{\Longleftrightarrow} & \vdash \boxed{\mathfrak{G}} \\
 & \overset{\text{insertion}}{\Longrightarrow} & \vdash \boxed{\mathfrak{G} \ \boxed{\mathfrak{G}'}} \\
 & \overset{\text{Ded.Theorem}}{\Longleftrightarrow} & \mathfrak{G} \vdash \mathfrak{G}' \\
 & \overset{\text{Def.}\vdash}{\Longrightarrow} & \mathfrak{H} \vdash \mathfrak{G}'
\end{array}$$

□

Lemma 6.12.

Let $\mathfrak{G}$, $\mathfrak{G}_1$ and $\mathfrak{G}_2$ be concept graphs and let $\mathfrak{H}$ be a set of concept graphs.

$$\begin{array}{rcll}
\mathfrak{H} \vdash \mathfrak{G} & \Longleftrightarrow & \mathfrak{H} \cup \{\boxed{\mathfrak{G}}\} \vdash \boxed{\quad} & , \\
\mathfrak{H} \vdash \boxed{\mathfrak{G}} & \Longleftrightarrow & \mathfrak{H} \cup \{\mathfrak{G}\} \vdash \boxed{\quad}, & , \\
\mathfrak{G}_1 \vdash \mathfrak{G}_2 & \Longleftrightarrow & \mathfrak{G}_1 \ \boxed{\mathfrak{G}_2} \vdash \boxed{\quad} & \textit{and} \\
\mathfrak{G}_1 \vdash \boxed{\mathfrak{G}_2} & \Longleftrightarrow & \mathfrak{G}_1 \ \mathfrak{G}_2 \vdash \boxed{\quad} & .
\end{array}$$

[1] Peirce treated existential graphs as *judgements*, i.e., as propositions which are asserted to be true in some context. A 'graph' which is never true in any context cannot be asserted and is therefore in Peirce's view not a graph. For this reason Peirce called the sketched graph (and all equivalent graphs) 'pseudograph' (see [42]).

Proof of the first equivalence: We have

$\mathfrak{H} \vdash \mathfrak{G}$

$\overset{\text{Def.}\vdash}{\iff}$ there are $\mathfrak{G}_1, \dots, \mathfrak{G}_n \in \mathfrak{H}$ with $\mathfrak{G}_1 \dots \mathfrak{G}_n \vdash \mathfrak{G}$

$\overset{\text{Ded.Theorem}}{\iff}$ there are $\mathfrak{G}_1, \dots, \mathfrak{G}_n \in \mathfrak{H}$ with $\vdash \boxed{\mathfrak{G}_1 \dots \mathfrak{G}_n \boxed{\mathfrak{G}}}$

$\overset{\text{double cut}}{\iff}$ there are $\mathfrak{G}_1, \dots, \mathfrak{G}_n \in \mathfrak{H}$ with $\vdash \boxed{\mathfrak{G}_1 \dots \mathfrak{G}_n \boxed{\mathfrak{G}} \ \boxed{\boxed{}}}$

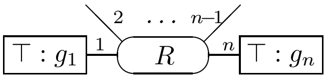
$\overset{\text{Ded.Theorem}}{\iff}$ there are $\mathfrak{G}_1, \dots, \mathfrak{G}_n \in \mathfrak{H}$ with $\mathfrak{G}_1 \dots \mathfrak{G}_n \boxed{\mathfrak{G}} \vdash \boxed{}$

$\overset{\text{Def.}\vdash}{\iff}$ $\mathfrak{H} \cup \{\boxed{\mathfrak{G}}\} \vdash \boxed{}$

The second equivalence is shown analogously. The third and fouth equivalence are immediate consequences of the first two equivalences. □

Definition 6.13 (Atomic Graphs).

A concept graph $\mathfrak{G}$ is called atomic*, if it has the form*

- $\boxed{P : g}$ *for $P \in \mathcal{C}$ and $g \in \mathcal{G}$ (this is called* singleton graph*) or*
- 

for $R \in \mathcal{R}_k$ and $g_1, \dots, g_n \in \mathcal{G}$ (this is called a star*).*

As we consider finite alphabets, there are (up to isomorphism) only finitely many atomic concept graphs. A concept a graph in anti-normal-form is composed of atomic graphs and cuts. More formally:

Lemma 6.14 (Iterative Construction of Atomic Graphs).

- *Every atomic concept graph is in anti-normal-form.*
- *If $\mathfrak{G}$ and $\mathfrak{G}'$ are two concept graphs in anti-normal-form, then both the juxtaposition $\mathfrak{G}\ \mathfrak{G}'$ and the negation $\boxed{\mathfrak{G}}$ are in anti-normal-form.*

Furthermore every graph in anti-normal-form can be composed this way.

Without proof.

Lemma 6.15 (Maximal Consistent Sets of Graphs).

For a maximal (with respect to $\subseteq$) consistent set of concept graphs $\mathfrak{H}$ we have:

1. *Either $\mathfrak{H} \vdash \mathfrak{G}$ or $\mathfrak{H} \vdash \boxed{\mathfrak{G}}$ for each concept graph $\mathfrak{G}$.*
2. *$\mathfrak{H} \vdash \mathfrak{G} \iff \mathfrak{G} \in \mathfrak{H}$ for each concept graph $\mathfrak{G}$.*

Proof: It is easy to see that $\mathfrak{G}\,\boxed{\mathfrak{G}} \vdash \boxed{\phantom{\mathfrak{G}}}$. Hence if $\mathfrak{H}$ is consistent, $\mathfrak{H} \vdash \mathfrak{G}$ and $\mathfrak{H} \vdash \boxed{\mathfrak{G}}$ cannot hold both. Now we can prove both propositions of this lemma:

1. Assume $\mathfrak{H} \nvdash \mathfrak{G}$ for a concept graph $\mathfrak{G}$. Lem. 6.12 yields that $\mathfrak{H} \cup \{\boxed{\mathfrak{G}}\}$ is consistent. As $\mathfrak{H}$ is maximal, we conclude that $\boxed{\mathfrak{G}} \in \mathfrak{H}$, hence we have $\mathfrak{H} \vdash \boxed{\mathfrak{G}}$.
2. Let $\mathfrak{H} \vdash \mathfrak{G}$. As $\mathfrak{H}$ is consistent, we get that $\mathfrak{H} \nvdash \boxed{\mathfrak{G}}$. So Lem. 6.12 yields that $\mathfrak{H} \cup \{\mathfrak{G}\}$ is consistent. As $\mathfrak{H}$ is maximal, we conclude $\mathfrak{G} \in \mathfrak{H}$. □

Consistent sets of concept graphs can be extended to maximal consistent sets of concept graphs. This is done as usual in logic.

Lemma 6.16 (Existence of Maximal Consistent Sets of Graphs).

Let $\mathfrak{H}$ be a consistent set of concept graphs. Then there is a maximal set $\mathfrak{H}'$ of concept graphs with $\mathfrak{H}' \supseteq \mathfrak{H}$.

Proof: Let $\mathfrak{G}_1, \mathfrak{G}_2, \mathfrak{G}_3, \ldots$ be an enumeration of all concept graphs.[2] We define inductively a set of graphs $\mathfrak{H}_i$ for each $i \in \mathbb{N}$. We start with setting $\mathfrak{H}_1 := \mathfrak{H}$. Assume now that $\mathfrak{H}_i \supseteq \mathfrak{H}$ is defined and consistent.

If $\mathfrak{H}_i \vdash \boxed{\mathfrak{G}_i}$ does not hold, then $\mathfrak{H}_{i+1} := \mathfrak{H}_i \cup \{\mathfrak{G}_i\}$ is consistent due to Lem. 6.12.

Otherwise, if $\mathfrak{H}_i \vdash \boxed{\mathfrak{G}_i}$ holds, then $\mathfrak{H}_{i+1} := \mathfrak{H}_i \cup \{\,\boxed{\mathfrak{G}_i}\,\}$ is consistent.

Now $\mathfrak{H}' := \bigcup_{n \in \mathbb{N}} \mathfrak{H}'_n$ is obviously a consistent maximal graph set with $\mathfrak{H}' \supseteq \mathfrak{H}$.
□

Maximal consistent set of graphs have canonically given models.

Theorem 6.17 (A Maximal Consistent Sets of Graphs has a Model).

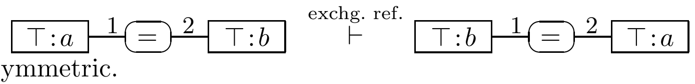
Let $\mathfrak{H}$ be a maximal consistent set of nonexistential concept graphs with cuts. Then there exists a canonically given contextual structure $(\vec{\mathbb{K}}, \lambda)$ such that $(\vec{\mathbb{K}}, \lambda) \models \mathfrak{G}$ for each graph $\mathfrak{G} \in \mathfrak{H}$.

Proof: Let $\varTheta := \{(a,b) \in \mathcal{G} \times \mathcal{G} \mid \mathfrak{H} \vdash \boxed{\top : a} - (=) - \boxed{\top : b}\ \}$

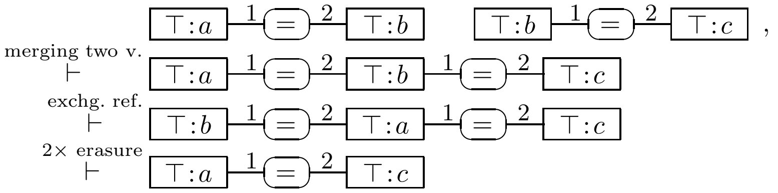
First we show that $\varTheta$ is an equivalence relation:

- $\top$-insertion and identity-insertion yield $\vdash \boxed{\top : a} \overset{1}{-} (=) \overset{2}{-} \boxed{\top : a}$, thus $\varTheta$ is reflexive.

[2] Remember that we assumed that $\mathcal{A}$ is finite, hence we have only countably many concept graphs (more precisely: countably many isomorphism-classes of concept graphs). If we allowed infinite alphabets, the proof could be carried out with an application of the prim ideal theorem or of (the stronger) Zorn's Lemma.

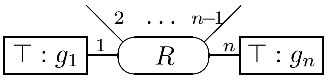
- We have  ,

hence Θ is symmetric.
- We have



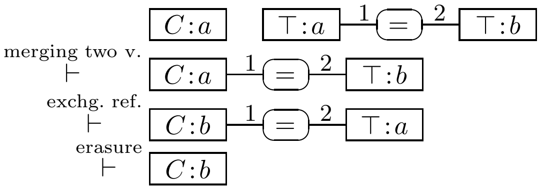
hence Θ is transitive.

Now we define a power context family $\vec{\mathbb{K}} := ((G_k, M_k, I_k))_{i=0,\ldots,n}$ (where n is the maximal arity of the relation-names) as follows:

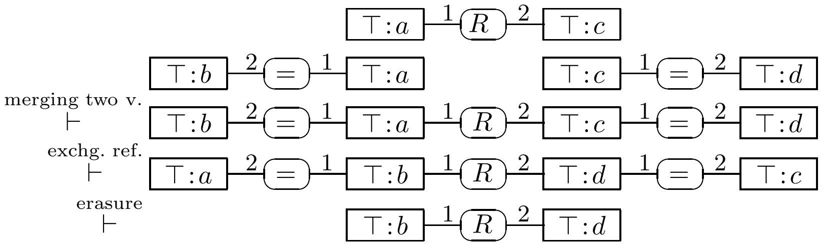
- $G_0 := \{[g]\Theta \mid g \in \mathcal{G}\}$, $G_k := {G_0}^k$, $M_0 := \mathcal{C}$, $M_k := \mathcal{R}_k$,
- $[g]\Theta\, I_0\, C \quad :\Longleftrightarrow \quad \mathfrak{H} \vdash \boxed{C : g}$,
- $([g_1]\Theta, \ldots, [g_k]\Theta)\, I_k\, R_k \quad :\Longleftrightarrow \quad \mathfrak{H} \vdash$  .

To see that the relations $I_0, I_1, \ldots, I_n$ are well defined, we have to prove that Θ is a congruence relation. This is done next.

- Θ is a congruence relation with respect to the concept names:



- Θ is a congruence relation with respect to dyadic relation names (the proof for relation names of other arities is done analogously):



We have shown so far that $\vec{\mathbb{K}}$ is a well defined power context family.

Now we define canonically

$$\lambda_{\mathcal{G}} : \begin{cases} \mathcal{G} \to G_0 \\ g \mapsto [g]\Theta \end{cases}, \lambda_{\mathcal{C}} : \begin{cases} \mathcal{C} \to \underline{\mathfrak{B}}(\mathbb{K}_0) \\ C \mapsto (C^{I_0}, C^{I_0 I_0}) \end{cases}, \lambda_{\mathcal{R}} : \begin{cases} \mathcal{R} \to \mathfrak{R}_{\vec{\mathbb{K}}} \\ R \in \mathcal{R}_k \mapsto (R^{I_k}, R^{I_k I_k}) \end{cases}$$

We have to show that $\lambda := \lambda_{\mathcal{G}} \,\dot{\cup}\, \lambda_{\mathcal{C}} \,\dot{\cup}\, \lambda_{\mathcal{R}}$ is a $\vec{\mathbb{K}}$-interpretation of $\mathcal{A}$, i.e., we have to show the conditions in Def. 4.3.

Let $C_1, C_2 \in \mathcal{C}$ be two concept names with $C_1 \leq C_2$. Let $[g]\Theta \in C_1^{I_0}$, i.e., we have $\mathfrak{H} \vdash \boxed{C_1 : g}$. By generalization we get $\mathfrak{H} \vdash \boxed{C_2 : g}$. This yields $C_1^{I_0} \subseteq C_2^{I_0}$, hence $\lambda_{\mathcal{C}}$ is order-preserving. It is analogously shown that $\lambda_{\mathcal{R}}$ is order-preserving. Using the $\top$-rule yields $\vdash \boxed{\top : g}$ for every $g \in \mathcal{G}$, hence $\lambda_{\mathcal{G}}(\top) = (G_0, G_0^{I_0}) = \top$. Finally we have

$$\begin{aligned} [g_1]\Theta = [g_2]\Theta &\overset{\text{Def. } \Theta}{\Longleftrightarrow} \mathfrak{H} \vdash \boxed{\top : g_1} \!-\!\!\left(\,=\,\right)\!\!-\! \boxed{\top : g_2} \\ &\overset{\text{Def. } I_2}{\Longleftrightarrow} ([g_1]\Theta, [g_2]\Theta) \in \text{Ext}(\lambda_{\mathcal{R}}(\doteq)) \end{aligned}$$

So λ is a $\vec{\mathbb{K}}$-interpretation of $\mathcal{A}$.

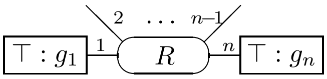
Now we show

$$(\vec{\mathbb{K}}, \lambda) \models \mathfrak{G}' \quad \Longleftrightarrow \quad \mathfrak{H} \vdash \mathfrak{G}' \tag{6.1}$$

for each concept graph $\mathfrak{G}'$ which is in anti-normal-form. The proof is done by induction, using Lem. 6.14. For the induction, we distinguish the following cases:

- $\mathfrak{G}' := \boxed{C : g}$ for a $C \in \mathcal{C}$ and a $g \in \mathcal{G}$. Then we have

$$\begin{aligned} (\vec{\mathbb{K}}, \lambda) \models \boxed{C : g} &\overset{\text{Def. } \models}{\Longleftrightarrow} \lambda_{\mathcal{G}}(g) \in Ext(\lambda_{\mathcal{C}}(C)) \\ &\overset{\text{Def. } \lambda}{\Longleftrightarrow} [g]\Theta \in C^{I_0} \\ &\Longleftrightarrow [g]\Theta I_0 C \\ &\overset{\text{Def. } I_0}{\Longleftrightarrow} \mathfrak{H} \vdash \boxed{C : g} \end{aligned}$$

The case $\mathfrak{G}' :=$  is done analogously.

- Now let $\mathfrak{G}'$ be a concept graph in anti-normal-form. Then we have

$$\begin{aligned} (\vec{\mathbb{K}}, \lambda) \models \left(\, \mathfrak{G}' \,\right) &\overset{\text{Def. } \models}{\Longleftrightarrow} (\vec{\mathbb{K}}, \lambda) \not\models \mathfrak{G}' \\ &\overset{\text{I.H.}}{\Longleftrightarrow} \mathfrak{H} \not\vdash \mathfrak{G}' \\ &\overset{\text{Lem. 6.15}}{\Longleftrightarrow} \mathfrak{H} \vdash \left(\, \mathfrak{G}' \,\right) \end{aligned}$$

- Finally let $\mathfrak{G}'$, $\mathfrak{G}''$ be two concept graphs in anti-normal-form. Then we have

$$\begin{aligned}(\vec{\mathbb{K}},\lambda) \models \mathfrak{G}' \quad \mathfrak{G}'' &\overset{\text{Def.} \models}{\iff} (\vec{\mathbb{K}},\lambda) \models \mathfrak{G}' \text{ and } (\vec{\mathbb{K}},\lambda) \models \mathfrak{G}'' \\ &\overset{\text{I.H.}}{\iff} \mathfrak{H} \vdash \mathfrak{G}' \text{ and } \mathfrak{H} \vdash \mathfrak{G}'' \\ &\overset{\text{Def.} \vdash}{\iff} \mathfrak{H} \vdash \mathfrak{G}' \quad \mathfrak{G}''\end{aligned}$$

This completes the proof of (6.1).

Now let $\mathfrak{G} \in \mathfrak{H}$. Lem. 5.8 yields that there is a graph $\mathfrak{G}'$ which is provably equivalent to $\mathfrak{G}$ and which is in anti-normal-form. We have $\mathfrak{H} \vdash \mathfrak{G}'$, hence (6.1) yields $(\vec{\mathbb{K}},\lambda) \models \mathfrak{G}'$. Now we conclude $(\vec{\mathbb{K}},\lambda) \models \mathfrak{G}$ from Thm. 6.9, and we are done. □

Now we are prepared to prove the completeness of the calculus.

Theorem 6.18 (Completeness of the Calculus).
Two nonexistential, simple concept graphs $\mathfrak{G}_1$, $\mathfrak{G}_2$ with cuts over $\mathcal{A}$ satisfy

$$\mathfrak{G}_1 \models \mathfrak{G}_2 \quad \Longrightarrow \quad \mathfrak{G}_1 \vdash \mathfrak{G}_2$$

Proof: Assume that $\mathfrak{G}_1 \vdash \mathfrak{G}_2$ does not hold. Then Cor. 6.12 yields that $\mathfrak{G}_1\ \boxed{\mathfrak{G}_2}$ is consistent. According to the last lemma, let be $\mathfrak{H}$ be a maximal consistent set of concept graphs which includes this graph. Due to Thm. 6.17, $\mathfrak{H}$ has a canonically given contextual structure $(\vec{\mathbb{K}},\lambda)$. This structure is in particular a model for $\mathfrak{G}_1\ \boxed{\mathfrak{G}_2}$. So $(\vec{\mathbb{K}},\lambda)$ is a model for $\mathfrak{G}_1$, but not for $\mathfrak{G}_2$, which is a contradiction to the assumption. □

6.3 Dependencies and Independencies

As we have already mentioned, concept graphs with cuts are based on existential graphs, and the first five rules of the calculus for concept graphs with cuts are a concept graph version of Peirce's calculus for existential graphs. But there are crucial differences between existential graphs and concept graphs.

First, existential graphs are composed of cuts, lines of identity and relation names. In contrast to the alphabet we use in concept graphs, the relation names which are used in existential graphs are not ordered. Furthermore we have names for objects in the alphabet for concept graphs. So we have a higher expressivity in concept graphs with cuts.

Second, the lines of identity of existential graphs are both used for existential quantification and for expressing identity. In concept graphs, two different syntactical elements are used for this purposes: Existential quantification is expressed by the generic marker ‘$*$’, and identity is expressed by identity links.

These facts have to be reflected by the calculus. For this reason we had to add further rules to the rules of Peirce. In this section we will investigate the logical relationships between the rules of Peirce and the new rules.

The new rules, except the rules ‘exchanging two references’ and ‘isomorphism’, are grouped in pairs (i.e., specialization/generalization, $\top$-erasure/$\top$-insertion, identity-erasure/identity-insertion, splitting a vertex/merging two vertices). A closer observation yields that these pairs satisfy the following conditions:

If one rule of a pair allows specific transformations in arbitrary *positive* contexts, then the other rule allows the inverse transformations in arbitrary *negative* contexts, and vice versa. For example, the generalization-rule allows a specific transformation in positive contexts: The concept names of vertices in that context, respectively the relation names of edges in that context, may be generalized. The specialization-rule is in fact the inverse direction of the generalization-rule and may only be applied in negative contexts.

If one rule of a pair allows specific transformations in arbitrary – *positive or negative* – contexts, then the other rule allows the inverse transformations in arbitrary contexts, and vice versa. An example for one of these pairs are the rules $\top$-erasure/$\top$-insertion.

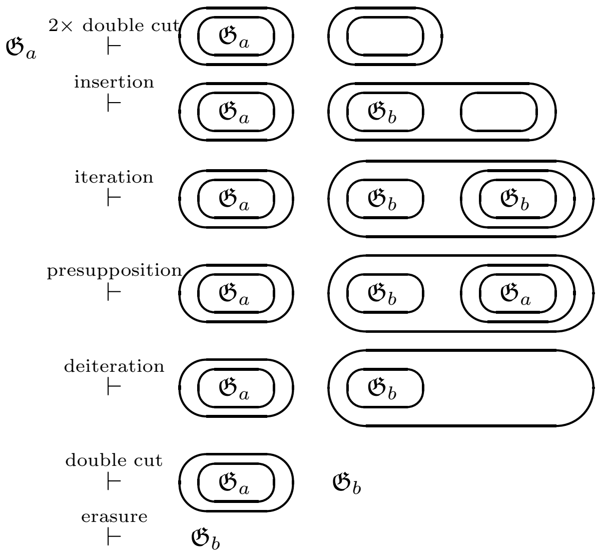
If a pair of rules satisfies one of the conditions above, then one of the rules may be removed, and the calculus is still complete. The fact that we may remove one rule of each pair of rules does not depend on the specific transformations of the rules. It is a general property of the calculus which can be shown by only using the first five rules (i.e., the rules of Peirce) of the calculus. This is subject of the next lemma.

Lemma 6.19 (Main Lemma for Dependencies in the Calculus). *Assume we have a calculus for concept graphs with cuts which contains the rules of Peirce (i.e., erasure, insertion, iteration, deiteration, and double cut). Let a further rule be given. Then we have:*

1. *If this rule allows specific transformations in arbitrary positive contexts, then we can derive that the inverse transformations may be carried out in arbitrary negative contexts.*
2. *If this rule allows specific transformations in arbitrary negative contexts, then we can derive that the inverse transformations may be carried out in arbitrary positive contexts.*
3. *If this rule allows specific transformations in arbitrary contexts, then we can derive that the inverse transformations may be carried out in arbitrary contexts.*

Proof:

1. Let a rule be given which allows us to carry out a specific transformation in arbitrary positive cuts. Let $\mathfrak{G}_a$, $\mathfrak{G}_b$ be two concept graphs such that $\mathfrak{G}_b$ is obtained from $\mathfrak{G}_a$ by applying the inverse transformation of the rule in a negative context. Our presupposition yields that $\boxed{\mathfrak{G}_b}$ may be replaced by $\boxed{\mathfrak{G}_a}$ in arbitrary positive contexts. Now 1) can be proven as follows:



2. Analogously to the last case.
3. Follows immediately from the 1) and 2). □

From this lemma, we can now immediately conclude the following proposition:

Proposition 6.20 (Main Proposition for Dependencies in the Calculus).

In the calculus for nonexistential concept graphs, we may

- *remove one of the rules 'specialization' and 'generalization', and*
- *remove one of the rules '⊤-erasure' and '⊤-insertion', and*
- *remove one of the rules 'identity-erasure' and 'identity-insertion', and*
- *remove one of the rules 'splitting a vertex' and 'merging two vertices',*

and the remaining calculus is still complete.

Without Proof.

As we have just shown, each of the pairs of rules specialization/generalization, $\top$-erasure/ $\top$-insertion, identity-erasure/identity-insertion, splitting a vertex/merging two vertices contains redundancy in the following sense: if one of the two rules of a pair is removed from the calculus, the calculus is still complete. However, in order to keep the calculus symmetric (like the calculus of Peirce — see Chap. 14.4), added in every case both rules to it.

In the following we will show that removing *both* rules of a pair simultaneously yields a strictly weaker calculus, i.e., the remaining calculus is not complete anymore (with respect to the entailment relation $\models$). The same holds if the rule 'exchanging two references' is removed. As we will show a couple of independency-results like this, we first provide a short description on how the proofs of these independency-results are carried out.

When we remove one or more rules from our calculus, the new, restricted calculus yields a new derivability relation $\vdash'$. For $\vdash'$ we will construct an alphabet $\mathcal{A}$ and a new semantical entailment relation $\models'$ such that the restricted calculus is sound with respect to $\models'$, i.e.:

$$\text{For all graphs } \mathfrak{G}, \mathfrak{G}' \text{ over } \mathcal{A} \text{ we have: } \mathfrak{G} \vdash' \mathfrak{G}' \text{ yields } \mathfrak{G} \models' \mathfrak{G}'$$

Then we will provide two graphs $\mathfrak{G}_1$, $\mathfrak{G}_2$ over $\mathcal{A}$ such that we have $\mathfrak{G}_1 \models \mathfrak{G}_2$, but $\mathfrak{G}_1 \not\models' \mathfrak{G}_2$.[3] From $\mathfrak{G}_1 \not\models' \mathfrak{G}_2$ we conclude $\mathfrak{G}_1 \not\vdash' \mathfrak{G}_2$. Then we have $\mathfrak{G}_1 \models \mathfrak{G}_2$ and $\mathfrak{G}_1 \not\vdash' \mathfrak{G}_2$, which shows that the new, restricted calculus is not complete with respect to the usual semantical entailment relation $\models$.

The first result we will prove in this manner is for the rules 'generalization' and 'specialization'.

Proposition 6.21 (Non-Redundancy of General./Special.).

The pair 'generalization' and 'specialization' cannot be derived from the remaining rules of the calculus.

Proof: Let $\vdash'$ be the derivability relation that the calculus without the rules 'generalization' and 'specialization' yields. We have to show $\vdash \neq \vdash'$.

Let $\mathcal{A} := (\{g\}, \{P, Q, \top\}, \{\doteq\})$ with $P \leq Q$. Now a new semantical entailment relation $\models'$ is introduced.In order to do this, the definition of the semantical entailment relation $\models$ (see Def. 4.4) is slightly changed:

$(\vec{\mathbb{K}}, \lambda) \models' \mathfrak{G}[c] :\Longleftrightarrow$

- $\kappa(v) \in \{P, \top\}$ for each $v \in V \cap area(c)$ (new vertex condition)
- $\lambda_{\mathcal{G}}(\rho(e)) \in Ext(\lambda_{\mathcal{R}}(\kappa(e)))$ for each $e \in E \cap area(c)$ (edge condition)
- $(\vec{\mathbb{K}}, \lambda) \not\models' \mathfrak{G}[c']$ for each $c' \in Cut \cap area(c)$ (cut condition)

With this definition $(\vec{\mathbb{K}}, \lambda) \models' \mathfrak{G}$ and $\mathfrak{G} \models' \mathfrak{G}'$ are defined as usual.

[3] As we have to show $\mathfrak{G}_1 \not\models' \mathfrak{G}_2$ it was sufficient to consider only one alphabet $\mathcal{A}$.

This is the old definition of $\models$ except the new vertex condition. Now it is easy to see that the rules of Peirce (i.e., erasure, insertion, iteration, deiteration and double-cut) are sound with respect to $\models'$ (the proofs which show that these rules are sound with respect to $\models$ work for $\models'$ as well). The soundness of the isomorphism-rule is obvious. The identity-erasure-rule and identity-insertion-rule are still sound because the interpretation of the relation-name $\doteq$ did not change. The new vertex condition does not change the interpretation of the concept name $\top$, either, hence the rules 'identity-erasure' and 'identity-insertion' as well as the rules 'merging two vertices' and 'splitting a vertex' are still sound.

Now let $\mathfrak{G}_1 := \boxed{P : g}$ and $\mathfrak{G}_2 := \boxed{Q : g}$. We have $\mathfrak{G}_1 \not\models' \mathfrak{G}_2$, but $\mathfrak{G}_1 \models \mathfrak{G}_2$. Thus, $\vdash'$ is not complete with respect to $\models$. □

We proceed with the rules '$\top$-erasure' and '$\top$-insertion'.

Proposition 6.22 (Non-Redundancy of $\top$-Erasure/Insertion).

The pair '$\top$-erasure' and '$\top$-insertion' cannot be derived from the remaining rules of the calculus.

Proof: Let $\vdash'$ be the derivability relation that the calculus without rules '$\top$-erasure' and '$\top$-insertion' yields. Let $\mathcal{A} := (\mathcal{G}, \mathcal{C}, \mathcal{R})$ be an arbitrary alphabet with $g \in \mathcal{G}$ (in particular we have $\mathcal{G} \neq \emptyset$). Again we slightly change the usual semantical entailment relation $\models$ in order to construct a new semantical entailment relation $\models'$. We set:

$(\vec{\mathbb{K}}, \lambda) \models' \mathfrak{G}[c] :\Longleftrightarrow$

- $V \cap area(c) = \emptyset$ (new vertex condition)
- $\lambda_{\mathcal{G}}(\rho(e)) \in Ext(\lambda_{\mathcal{R}}(\kappa(e)))$ for each $e \in E \cap area(c)$ (edge condition)
- $(\vec{\mathbb{K}}, \lambda) \not\models' \mathfrak{G}[c']$ for each $c' \in Cut \cap area(c)$ (cut condition)

For all rules except the rules '$\top$-erasure', '$\top$-insertion', 'merging two vertices' and 'splitting a vertex', the proofs of their soundness with respect to $\models$ work for $\models'$ as well, i.e., all these rules are sound with respect to $\models'$. Now let $\mathfrak{G}$ and $\mathfrak{G}'$ be two graphs with v_1, v_2 in V such that $\mathfrak{G}'$ is derived from $\mathfrak{G}$ by merging v_2 into v_1, and let $(\vec{\mathbb{K}}, \lambda)$ be a contextual structure over $\mathcal{A}$. It is easy to see that $\mathfrak{G}$ and $\mathfrak{G}'$ are isomorphic except for $ctx(v_1)$, and we have $(\vec{\mathbb{K}}, \lambda) \not\models' \mathfrak{G}[ctx(v_1)]$ as well as $(\vec{\mathbb{K}}, \lambda) \not\models' \mathfrak{G}'[ctx(v_1)]$. Hence Thm. 6.1 can be applied.[4] So we get $(\vec{\mathbb{K}}, \lambda) \models' \mathfrak{G} \Longleftrightarrow (\vec{\mathbb{K}}, \lambda) \models' \mathfrak{G}'$, i.e., the rules 'merging two vertices' and 'splitting a vertex' are sound with respect to $\models'$, too.

Now let $\mathfrak{G}_1$ be the empty graph and $\mathfrak{G}_2 := \boxed{\top : g}$. Then we have $\mathfrak{G}_1 \not\models' \mathfrak{G}_2$, but $\mathfrak{G}_1 \models \mathfrak{G}_2$. Thus, $\vdash'$ is not complete with respect to $\models$. □

[4] It is easy to see that Thm. 6.1 can be proven for $\models'$ as well.

Proposition 6.23 (Non-Redundancy of Id-Erasure/Insertion).

The pair 'identity-erasure' and 'identity-insertion' cannot be derived from the remaining rules of the calculus.

Proof: Let $\vdash'$ be the calculus without the rules 'identity-erasure' and 'identity-insertion'. Let $\mathcal{A} := (\{g\}, \{P, Q, \top\}, \{\dot{=}\})$ be an alphabet with incomparable concept-names P, Q. Again, a new semantical entailment relation $\models'$ is introduced. It is defined as follows:

$(\vec{\mathbb{K}}, \lambda) \models' \mathfrak{G}[c] :\Longleftrightarrow$

- $\lambda_{\mathcal{G}}(\rho(v)) \in Ext(\lambda_{\mathcal{C}}(\kappa(v)))$ for each $v \in V \cap area(c)$ (vertex condition)
- $\lambda_{\mathcal{G}}(\rho(e)) \in Ext(\lambda_{\mathcal{R}}(\kappa(e)))$ for each $e \in E^{nonid} \cap area(c)$ (edge condition)
- $\lambda_{\mathcal{G}}(\rho(v_1)) \neq \lambda_{\mathcal{G}}(\rho(v_2))$ for each $id = (v_1, v_2) \in E^{id} \cap area(c)$ (new identity condition)
- $(\vec{\mathbb{K}}, \lambda) \not\models' \mathfrak{G}[c']$ for each $c' \in Cut \cap area(c)$ (cut condition)

Again this is the old definition of $\models$ except the new identity condition. Similar to the preceding proofs it is easy to see that all rules except 'exchanging references', 'identity-erasure' and 'identity-insertion' are sound with respect to $\models'$. As $\mathcal{A}$ has only one object-name, the rule 'exchanging references' has no syntactical effect at all and is therefore trivially sound. Hence $\vdash'$ is sound with respect to $\models'$.

But the graphs $\mathfrak{G}_1 := \boxed{P : g}\ \boxed{Q : g}$ and $\mathfrak{G}_2 := \boxed{P : g}\!-\!(=)\!-\!\boxed{Q : g}$ satisfy $\mathfrak{G}_1 \not\models' \mathfrak{G}_2$, but $\mathfrak{G}_1 \models \mathfrak{G}_2$. Thus, $\vdash'$ is not complete with respect to $\models$. □

Proposition 6.24 (Non-Redundancy of Exchanging References).

The rule 'exchanging references' cannot be derived from the remaining rules of the calculus.

Proof: Let $\vdash'$ be the calculus without the rule 'exchanging references'. Let $\mathcal{A} := (\{1, 2\}, \{\top\}, \{\dot{=}\})$. Again we have to provide a new semantical entailment relation $\models'$. As we want to construct two graphs $\mathfrak{G}_1, \mathfrak{G}_2$ with $\mathfrak{G}_1 \not\models' \mathfrak{G}_2$, it is sufficient to consider a specific contextual structure $(\vec{\mathbb{K}}, \lambda)$. We consider the contextual structure $(\vec{\mathbb{K}}, \lambda)$ with $G_0 := \{1, 2\}$, $M_1 := \{\top\}$ and $M_2 := \{\dot{=}\}$. Of course we set $\lambda_{\mathcal{G}}(1) := 1$ and $\lambda_{\mathcal{G}}(1) := 2$. According to Def. 4.3, $\lambda_{\mathcal{C}}(\top)$ and $\lambda_{\mathcal{R}}(\dot{=})$ have to defined as usual. But, similar to the proof of Proposition 6.23, we change the interpretation of identity-links in the definition of the new relation $\models'$. On the set G_0, we have the canonically given ordering of the natural numbers (i.e. $1 \leq 1$, $2 \leq 2$, $1 \leq 2$). We will interpret the sign $\dot{=}$ by this ordering. We set:

$(\vec{\mathbb{K}}, \lambda) \models' \mathfrak{G}[c] :\Longleftrightarrow$

- $\lambda_{\mathcal{G}}(\rho(v)) \in Ext(\lambda_{\mathcal{C}}(\kappa(v)))$ for each $v \in V \cap area(c)$ (vertex condition)
- $\lambda_{\mathcal{G}}(\rho(e)) \in Ext(\lambda_{\mathcal{R}}(\kappa(e)))$ for each $e \in E^{nonid} \cap area(c)$ (edge condition)
- $\lambda_{\mathcal{G}}(\rho(v_1)) \leq \lambda_{\mathcal{G}}(\rho(v_2))$ for each $e = (v_1, v_2) \in E^{id} \cap area(c)$ (new identity condition)
- $(\vec{\mathbb{K}}, \lambda) \not\models' \mathfrak{G}[c']$ for each $c' \in Cut \cap area(c)$ (cut condition)

Again, the proofs for the soundness of the rules, with the exception of the rules 'identity-erasure', 'identity-insertion', 'merging two vertices', 'splitting a vertex' and 'exchanging references', work for $\models'$ as well. The identity-erasure-rule and identity-insertion-rule are still sound (in our chosen model) because the relation $\leq$ entails the identity-relation. Hence, any identity-link between two vertices which have the same reference is still a redundant information and may be therefore erased or inserted. The rules 'merging two vertices' and 'splitting a vertex' are still sound because they do not have any effect on identity-links.

But obviously, for $\mathfrak{G}_1 := \boxed{\top{:}1}\!-\!^{1}\!(=)\!^{2}\!-\!\boxed{\top{:}2}$ and $\mathfrak{G}_2 := \boxed{\top{:}2}\!-\!^{1}\!(=)\!^{2}\!-\!\boxed{\top{:}1}$, we have $\mathfrak{G}_1 \not\models' \mathfrak{G}_2$, but $\mathfrak{G}_1 \models \mathfrak{G}_2$. Thus, $\vdash'$ is not complete with respect to $\models$. □

Proposition 6.25 (Non-Redundancy of Merg./Split. Vertices). *The pair 'merging two vertices' and 'splitting a vertex' cannot be derived from the remaining rules of the calculus.*

Proof: Let $\vdash'$ be the calculus without the the rules 'merging two vertices' and 'splitting a vertex'. We set $\mathcal{A} := (\{g\}, \{P, \top\}, \{R, \doteq\})$ and define the new semantical entailment relation $\models'$ as follows:

$(\vec{\mathbb{K}}, \lambda) \models' \mathfrak{G}[c] :\Longleftrightarrow$

- for each $v \in V \cap area(c)$ with $\kappa(v) = P$ there is no edge $e \in E$ which is incident with v (new vertex condition)
- $\lambda_{\mathcal{G}}(\rho(e)) \in Ext(\lambda_{\mathcal{R}}(\kappa(e)))$ for each $e \in E \cap area(c)$ (edge condition)
- $(\vec{\mathbb{K}}, \lambda) \not\models' \mathfrak{G}[c']$ for each $c' \in Cut \cap area(c)$ (cut condition)

Again it is easy to see that all rules except 'merging two vertices' and 'splitting a vertex' are sound with respect to $\models'$.

But obviously, for $\mathfrak{G}_1 := \boxed{P : g}\quad\boxed{\top : g}\!-\!(R)$ and $\mathfrak{G}_2 := \boxed{P : g}\!-\!(R)$, we have $\mathfrak{G}_1 \not\models' \mathfrak{G}_2$, but $\mathfrak{G}_1 \models \mathfrak{G}_2$. Thus, $\vdash'$ is not complete with respect to $\models$. □

We have seen that if we remove the rule 'exchanging references' or both rules of a pair like specialization/generalization etc. simultaneously, the calculus is strictly weakened (i.e., it is not complete anymore). From this we conclude in particular that a concept graph version of Peirce's calculus for existential graphs is not sufficient to reflect the higher expressivity and the different syntactical structure of concept graphs with cuts.

7 Overview for Beta

In the part Beta of this treatise, we add generic markers to concept graphs with cuts, i.e., we now consider *existential* instead of nonexistential concept graphs with cuts. This is a crucial step: The difference between nonexistential and existential concept graphs is like the difference between FOL without and FOL with variables. With generic markers, we have the possibility to speak about indefinite objects, and we can draw propositions which range over our universe of discourse.

In fact the existential concept graphs are in a specific manner equivalent to FOL. In order to investigate and discuss this equivalence, we introduce a translation of CG to FOL (and vice versa). This is a main difference between the part Alpha and the part Beta of this treatise and leads (compared to the part Alpha) to extended investigations we have to carry out. We have to show that the contextual semantics of concept graphs, the calculus on concept graphs, and the translation of concept graphs to FOL fit together. This will be subject of this part of this treatise.

We start the part Beta with a chapter on FOL. As every book on logic provides its own style of FOL, this chapter is necessary. We provide all definitions that are relevant for Beta, but we do not give any proofs. For this we refer to the wide range of literature on FOL.

In the following chapter, we provide the contextual semantics for concept graphs as well as the syntactical translations from and to FOL (i.e., we provide a mapping Φ from CG to FOL and a mapping Ψ in the opposite direction). Of course, a translation of concept graphs yields the classical semantics of FOL for concept graphs in an indirect way, too. We will show that these two semantics are equivalent.

In Chap. 10 we present the calculus for existential concept graphs. This calculus is an extension of the calculus for nonexistential graphs in the following sense: It consists of the same set of rules, but most rules are slightly extended in order to respect the generic markers. Furthermore we show that the calculus is sound. The idea for the proof is the same as in the part Alpha. In fact, in order to prove the soundness of the Beta-rules, we can refer to the corresponding proofs in the part Alpha for most of the rules. But some of the rules (esp. iteration and deiteration) require new proofs of their soundness.

F. Dau: The Logic System of Concept Graphs with Negation, LNAI 2892, pp. 81-82, 2003.

In Chap. 11 we start to unfold out that the mappings Φ and Ψ can be considered as translations. In order to do this, we show the following: Mapping a formula f to a concept graph, and mapping this concept graph back to a formula yields a formula which is provably equivalent to f (but it will be usually different to f). This is done in Sect. 11.1. We get an analogous result if we start with a concept graph, which is subject of Sect. 11.2. Finally we show in Sect. 11.3 that Ψ respects the derivability relation $\vdash$. This is done by proving that Ψ respects every rule of the calculus on FOL, i.e., every rule of FOL carries over to a proof in CG. It turns out that this is the most extensive proof of this treatise.

As the rules of the calculus for concept graphs are very powerful, the proof of the corresponding proposition for concept graphs (i.e., to prove that Φ respects $\vdash$) would be much more complicated. For this reason we do not show this proposition directly. Nevertheless it will turn out in Chap. 12 that we will gain this proposition indirectly in a very easy way.

In fact we put together six main results, which we have proven in the preceding chapter, in this chapter. Using these results, it is now very easy to show the full semantical and syntactical equivalence of concept graphs and FOL. In particular it turns out that Φ and Ψ are mutually inverse isomorphisms between the quasiordered sets of concept graphs and first order logic, and we can show very easily that the calculus for existential concept graphs is sound and complete. This is subject of Sect. 12.1. On the other hand we show in Sect. 12.2 that it was *necessary* to prove *all* six results we have mentioned above: None of the six results can be proven from the remaining five in a trivial way.

We have already mentioned in the Alpha-part that for concept graphs without cuts, it is possible to assign to each graph a *standard model* which contains exactly the same information as the graph (and vice versa, we can assign to each model a *standard graph*). With standard models and standard graphs, reasoning can be carried over from graphs to models, and vice versa. This approach was introduced by Perdiger in her PhD-thesis (see [44]). In Chap. 13, we adopt and extend Predigers ideas for concept graphs without cuts.

8 First Order Logic

In this chapter we will present the style of first order logic (FOL) we will need in the rest of this treatise. We will consider first order predicate logic, that is first order logic with predicates, identity and object names, but without function names. Furthermore our logic is based on alphabets $\mathcal{A} := (\mathcal{G}, \mathcal{C}, \mathcal{R})$ as we use it for concept graphs. These alphabets have some properties which are usually not used in classical logic, namely

- we distinguish between concept names and unary relation names,
- the sets $\mathcal{C}$ and $\mathcal{R}$ are ordered, and
- the set $\mathcal{C}$ contains a greatest element $\top$ which stands for the concept which contains every object (of the respective context).

These properties have to be reflected in our version of first order logic, i.e., in our definition for models as well as in the calculus we will present. Nevertheless, for readers who are familiar with logic it is enough to have a glance at the definitions.

8.1 Syntax

We start with the definition of the well-formed formulas of FOL.

Definition 8.1 (Formulas).
The formulas of FOL over $\mathcal{A}$, the sets $Bound(f)$ of bound and $Free(f)$ of free variables of a formula f, and the set $Subform(f)$ of subformulas of f are inductively defined as follows:

1. *Each variable $\alpha \in Var$ and each object name $g \in \mathcal{G}$ is a* term.
   - $Free(\alpha) := \{\alpha\}$, $Free(g) := \emptyset$, $Bound(\alpha) := Bound(g) := \emptyset$.

F. Dau: The Logic System of Concept Graphs with Negation, LNAI 2892, pp. 83-91, 2003.

2. *If $C \in \mathcal{C}$ is a concept name and t is a term, then $f := C\underline{(}t\underline{)}$ is a formula.*[1]
   - $Free(f) := Free(t)$ *and* $Bound(f) := \emptyset$.
   - $Subform(f) := \{f\}$.
3. *If $R \in \mathcal{R}$ is a relation name with arity n and $t_1, \ldots, t_n$ are terms, then $f := R\underline{(}t_1\underline{,}\ldots\underline{,}t_n\underline{)}$ is a formula.*
   - $Free(f) := Free(t_1) \cup \ldots \cup Free(t_n)$, $Bound(f) := \emptyset$.
   - $Subform(f) := \{f\}$.
4. *If f' is a formula, then $f := \underline{\neg}f'$ is a formula.*
   - $Free(f) := Free(f')$, $Bound(f) := Bound(f')$.
   - $Subform(f) := Subform(f') \cup \{f\}$.
5. *If f_1 and f_2 are formulas, then $f := \underline{(}f_1\underline{\triangle}f_2\underline{)}$ is a formula.*
   - $Free(f) = Free(f_1) \cup Free(f_2)$, $Bound(f) = Bound(f_1) \cup Bound(f_2)$
   .
   - $Subform(f) := Subform(f_1) \cup Subform(f_2) \cup \{f\}$.
6. *If f' is a formula and α is a variable, then $f := \underline{\exists}\alpha\underline{.}f'$ is a formula.*
   - $Free(f) := Free(f')\backslash\{\alpha\}$, $Bound(f) := Bound(f') \cup \{\alpha\}$.
   - $Subform(f) := Subform(f') \cup \{f\}$.

If f is a formula with $Free(f) = \emptyset$, then f is said to be closed. *A closed formula is also called a* sentence.

For some dyadic relation names R, esp. for '$R = \doteq$', we will use the infix-notation instead of the prefix-notation, i.e., we will write $t_1 R t_2$ instead of $R(t_1, t_2)$ (in particular, we will write $t_1 \doteq t_2$ or $t_1 = t_2$ instead of $\doteq (t_1, t_2)$).

Keep in mind that we use an identity-relation in the formulas of FOL as well as in the metalanguage. We distinguish these two levels of identity by using the symbol '$\doteq$' for the identity on the syntactical level (in formulas as well as in graphs) and by using symbol '$=$' to denote the identity on the meta-level. In some cases we try to ease the reading by using different spaces around '$=$' and by using the symbol $\doteq$. For example, in '$f \ = \ x_3 \doteq x_5$', the first '$=$' is a metalevel sign, and the second '$\doteq$' is a sign in FOL. So the string '$f \ = \ x_3{\doteq}x_5$' means that f is the formula '$x_3 \doteq x_5$'.

[1] In this definition, we have underlined the signs which have to be understood literally. For example, when we write $f := R\underline{(}t_1\underline{,}\ldots\underline{,}t_n\underline{)}$, this means that the formula f is defined to be the following sequence of signs: We start with the relation name which is denoted by R. Please note that 'R' *is* not a relation name, but it is a metavariable which *stands for* a relation name. After R, we proceed with the sign '('. After that, we write down the term which is denoted by t_1 (thus, t_1 is a metavariable, too). We proceed with the sign ','. After that, we write down the term which is denoted by t_2, proceed with the sign ',', and so on until we write down the term is denoted by t_n. Finally, we write down the sign ')'. So, possible results for f are therefore '$faster(Yoyo, Garfield)$' or '$on(Yoyo, x_5)$' (if 'Yoyo' and 'Garfield' are object names in $\mathcal{G}$ and if 'faster' and 'on' are relation names in $\mathcal{G}$).

The remaining junctors (i.e., $\vee$, $\rightarrow$ and $\leftrightarrow$) and the remaining quantifier $\forall$ are defined as usual: We set

- $f_1 \vee f_2 := \neg(\neg f_1 \wedge \neg f_2)$,
- $f_1 \rightarrow f_2 := \neg(f_1 \wedge \neg f_2)$,
- $f_1 \leftrightarrow f_2 := \neg(f_1 \wedge \neg f_2) \wedge \neg(f_2 \wedge \neg f_1)$, and
- $\forall \alpha.f := \neg\exists\alpha.\neg f$.

This shall be read as an abbreviation: e.g. when we write $f_1 \rightarrow f_2$, we can replace this by $\neg(f_1 \wedge \neg f_2)$ to get a formula in our language.

Brackets are omitted or added to improve the readability of formulas. To avoid an overload of brackets, we agree that formulas which are composed with the dyadic junctor $\rightarrow$ are bracket from the right, e.g. $f_1 \rightarrow f_2 \rightarrow f_3$ has to read as $(f_1 \rightarrow (f_2 \rightarrow f_3))$. Furthermore we agree that quantifiers bind more strongly than binary junctors.

It will turn out later (especially in Def. 8.5) that it is important to distinguish between subformulas and *subformula occurences* of a formula f. For example, the formula $f := \top(x) \wedge \top(x)$ has only two subformulas, namely $\top(x)$ and f itself. But the subformula $\top(x)$ occurs twice in f, hence we have three subformula occurences in f (two times $\top(x)$ and f itself).

Defining subformula occurences is a rather technical problem, thus, we do not provide a definition for them, but we want to point out that it can be done (for a further discussion we refer to [12]). Of course, this is the same for terms: A term t can appear several times in a formula f, and all these appearences are called *occurences of* t *in* f.

Next we define substitutions.

Definition 8.2 (Substitutions).

Let $u \in \mathcal{G} \cup Var$ be a term and let $x_i \in Var$ be a variable. Furthermore, $t, t_1, t_2, \ldots$ denote terms and $f, f', f_1, f_2, \ldots$ denote formulas. We define the substitutions $t[u/x_i]$ and $f[u/x_i]$ inductively as follows:

1. *$x_j[u/x_i] := x_j$ for $x_j \in Var$ and $j \neq i$, $x_i[u/x_i] := u$ and $h[u/x_i] := h$ for $h \in \mathcal{G}$.*
2. *If $f := C(t)$, $C \in \mathcal{C}$, then $f[u/x_i] := C(t[u/x_i])$.*
3. *If $f := R(t_1, \ldots, t_n)$, $R \in \mathcal{R}_n$, then $f[u/x_i] := R(t_1[u/x_i], \ldots, t_n[u/x_i])$.*
4. *If $f := \neg f'$, then $f[u/x_i] := \neg f'[u/x_i]$.*
5. *If $f := f_1 \wedge f_2$, then $f[u/x_i] := f_1[u/x_i] \wedge f_2[u/x_i]$.*
6. *If $f := \exists x_j.f'$, $j \neq i$, then $f[u/x_i] := \exists x_j.f'[u/x_i]$.*
7. *If $f := \exists x_i.f'$, then $f[u/x_i] := \exists x_i.f'$.*

We say that $t[u/x_i]$ *is obtained from* t *by* substituting u for x_i in t*, and* $f[u/x_i]$ *is obtained from* f *by* substituting u for x_i in f*.*

A main difference between FOL and CG are the syntactical elements which are used to range over objects. In CG, only one sign, namely the generic marker '$*$', is used for this purpose[2]. In FOL we have a whole set Var of variables instead. All variables are tantamount. For this reason the next well-known definition is needed.

Definition 8.3 (α-Conversion of Formulas).

Let f *be a formula and let* $\exists\alpha.h$ *be a subformula of* f*. Let* β *be a (so called* fresh*) variable (i.e. we have* $\beta \notin Var(f)$*). Let* f' *be the formula that we get when we replace the subformula* $\exists\alpha.h$ *by* $\exists\beta.h[\beta/\alpha]$*. Then we say that we get* f' *from* f *by* renaming a bound variable *(this is in literature often called* α-conversion *of a formula).*

Example: Consider the formula

$$R(x) \wedge \exists x.S(x) \wedge \exists x.(R(x) \wedge \exists x.T(x))$$

If we replaced the first bound occurence of x – i.e., we consider the subformula $\exists x.S(x)$ – by the variable y, we get the formula

$$R(x) \wedge \exists y.S(y) \wedge \exists x.(R(x) \wedge \exists x.T(x))$$

In this formula, consider the subformula $\exists x.(R(x) \wedge \exists x.T(x))$. We replace x by u and get

$$R(x) \wedge \exists y.S(y) \wedge \exists u.(R(u) \wedge \exists x.T(x))$$

Finally, we replace the remaining bound variables x in $\exists x.T(x)$ by v and get

$$R(x) \wedge \exists y.S(y) \wedge \exists u.(R(u) \wedge \exists v.T(v))$$

The last formula has a form for which the proof of the equivalence of f and $\Phi(\Psi(f))$ (see Sect. 11.1) is simpler than for the general case. This form is captured by the following definition:

Definition 8.4 (Variable-Purified Formulas).

Let f *be a formula. If we have that* $Bound(f) \cap Free(f) = \emptyset$ *and if no variable is bound more than once (i.e., the string* $\exists\alpha$ *does not occur twice in* f *for any variable* α*) then* f *is said to be variable-purified.*

[2] The indexed generic markers $*_\alpha$ with $\alpha \in$ Var we have introduced could be understood as the 'free variables' of concept graphs. But in this treatise they are only used when we provide translations between FOL and CG. They are not used in the semantics for concept graphs, neither in the calculus.

Using α-conversion, we can transform each formula f by renaming bound variables into a formula f' which is variable-purified. It is well known that f and f' are equivalent (which will be defined in the next section).

In formulas of FOL, the set of all occurences of the negation sign '$\neg$' can be treated like the set of all cuts in a concept graphs. This motivates the next definition:

Definition 8.5 (Structure of Formulas). $Neg_f \cup \{\top_f\}$
Let $f \in FOL$ be a formula. We set

$$Neg_f := \{g \in FOL \mid \neg g \textit{ is a subformula occurence of } f\}$$

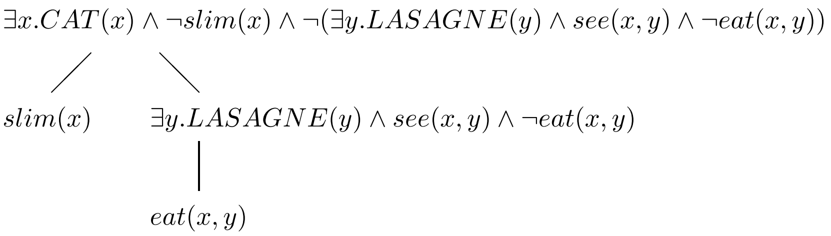
Furthermore we set $\top_f := f$. The set $Neg_f \cup \{\top_f\}$ is ordered by

$$g \leq h :\Longleftrightarrow g \textit{ is a subformula occurence of } h$$

The set Neg_f of a formula f is the counterpart of the set Cut of a graph $\mathfrak{G} := (V, E, \nu, \top, Cut, area, \kappa, \rho)$, and $Neg_f \cup \{\top_f\}$ is the counterpart of $Cut \cup \{\top\}$. To give an example, look at the following formula:

$$f := \exists x.(CAT(x) \wedge \neg slim(x) \wedge \neg(\exists y.LASAGNE(y) \wedge see(x,y) \wedge \neg eat(x,y)))$$

The ordered set $(Neg_f \cup \{\top_f\}, \leq)$ can be sketched as follows:

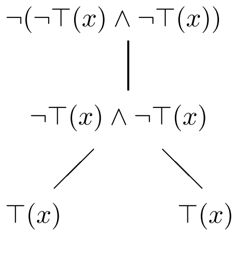

$\exists x.CAT(x) \wedge \neg slim(x) \wedge \neg(\exists y.LASAGNE(y) \wedge see(x,y) \wedge \neg eat(x,y))$
$slim(x)$
$\exists y.LASAGNE(y) \wedge see(x,y) \wedge \neg eat(x,y)$
$eat(x,y)$

Compare this with the order on the graph we have sketched in Fig. 2.3 on p. 34.

Another example is

$$g := \neg(\neg\top(x) \wedge \neg\top(x))$$

This example shows that is is important to discriminate between subformulas and subformula occurences. The ordered set $(Neg_g \cup \{\top_g\}, \leq)$ can be sketched as follows:


$\neg(\neg\top(x) \wedge \neg\top(x))$
$\neg\top(x) \wedge \neg\top(x)$
$\top(x)$
$\top(x)$

8.2 Semantics

In Chap. 4 and 9 we present the semantics for nonexistential resp. existential concept graphs. These semantics are based on power context families which serve as models for concept graphs and are therefore called *contextual.* The usual models for FOL-formulas are not power context families, but relational structures over $\mathcal{A}$. Classical relational structures are pairs $\mathcal{M} := (U, I)$, consisting of a *universe* (of discourse) U and a function $I := I_{\mathcal{G}} \dot{\cup} I_{\mathcal{R}}$ with $I_{\mathcal{G}} : \mathcal{G} \to U$ and $I_{\mathcal{R}} : \mathcal{R}_k \to \mathfrak{P}(U^k)$ for each k.

We adopt these relational structures for FOL as well. But as mentioned in the beginning of this chapter, our version of FOL is based on an alphabet with some properties which are usually not used in classical logic. These properties have to be respected by the models we consider. For this reason the usual notation of a relational structure is restricted as follows:

Definition 8.6 (Relational Structures).

A relational structure over $\mathcal{A}$ *is a pair* $\mathcal{M} := (U, I)$ *consisting of a nonempty* universe U *and a function* $I := I_{\mathcal{G}} \dot{\cup} I_{\mathcal{C}} \dot{\cup} I_{\mathcal{R}}$ *with* $I_{\mathcal{G}} : \mathcal{G} \to U$*,* $I_{\mathcal{C}} : \mathcal{C} \to \mathfrak{P}(U)$ *and* $I_{\mathcal{R}} : \mathcal{R}_k \to \mathfrak{P}(U^k)$ *for each* k *such that* $I_{\mathcal{C}}$ *and* $I_{\mathcal{R}}$ *are order-preserving, satisfy* $I_{\mathcal{C}}(\top) = U$*, and* $(u_1, u_2) \in I_{\mathcal{R}}(\doteq) \Leftrightarrow u_1 = u_2$ *for all* $u_1, u_2 \in U$*.*

The function I (the letter 'I' stands of course for 'interpretation') is the link between our language and the mathematical universe, i.e., it relates syntactical objects to mathematical entities. There are certain similarities between relational structures (U, I) and contextual structures $(\vec{\mathbb{K}}, \lambda)$ (in particular between the interpretation functions I and λ). This will be worked out in Sect. 9.3.

Now we define how formulas are evaluated in relational structures.

Definition 8.7 (Evaluation of Fourmulas in Structures).

Let $f \in FOL$ *be a formula and let* $\mathcal{M} = (U, I)$ *be a relational structure. A* valuation *val is a mapping* $val : Var \to U$ *which assigns to each variable an element of the universe* U*. Valuations are naturally extended to mappings* $val_I : Var \cup \mathcal{G} \to U$ *by* $val_I(g) := I_{\mathcal{G}}(g)$ *for each object name* $g \in \mathcal{G}$*. Now, inductively over the structure of formulas, we define* $\mathcal{M} \models_{val} f$ *as follows:*

1. *If* $C \in \mathcal{C}$ *and if* t *is a term, then we set* $\mathcal{M} \models_{val} C(t)$ *iff* $val_I(t) \in I_{\mathcal{C}}(C)$*.*
2. *If* $R \in \mathcal{R}_n$ *and if* $t_1, \ldots, t_n$ *are terms, then we set* $\mathcal{M} \models_{val} R(t_1, \ldots, t_n)$ *iff* $(val_I(t_1), \ldots, val_I(t_n)) \in I_{\mathcal{R}}(R)$*.*
3. *If* f' *is a formula and* $f := \neg f'$*, then we set* $\mathcal{M} \models_{val} f$ *iff* $\mathcal{M} \not\models_{val} f'$ *.*
4. *If* f_1 *and* f_2 *are formulas and* $f := f_1 \wedge f_2$*, then we set* $\mathcal{M} \models_{val} f$ *iff* $\mathcal{M} \models_{val} f_1$ *and* $\mathcal{M} \models_{val} f_2$ *.*

5. *If f' is a formula, x_i is a variable and $f := \exists x_i.f'$, then we set $\mathcal{M} \models_{val} f$ iff there is a valuation val' with $val'(x_j) = val(x_i)$ for each $i \neq j$ and $\mathcal{M} \models_{val'} f'$.*

If f is a formula and $\mathcal{M}$ is a relational structure such that $\mathcal{M} \models_{val} f$ holds for all valuations val, we write $\mathcal{M} \models f$. If F is a set of formulas and f is a formula such that each relational structure $\mathcal{M}$ with $\mathcal{M} \models g$ for all $g \in F$ satisfies $\mathcal{M} \models f$, we write $F \models f$. We abbreviate $\{g\} \models f$ by $g \models f$. Two formulas f, g with $f \models g$ and $g \models f$ are called semantically equivalent*.*

8.3 Calculus

In literature we find a huge amount of calculi for first order logic. For our purpose it makes no difference which calculus we use. We decided to choose a (so-called Hilbert-style) calculus in the fragment of FOL which is based on the junctors $\rightarrow$ and $\neg$ instead of $\wedge$ and $\neg$. This causes no troubles, as the following argumentation will show:

Remember that we can express the junctor $\rightarrow$ by means of the junctors $\wedge$ and $\neg$: We set $f_1 \rightarrow f_2 := \neg(f_1 \wedge \neg f_2)$ (and conversely, we can express the junctor $\wedge$ with the junctors $\rightarrow$ and $\neg$: We set $f_1 \wedge f_2 := \neg(f_1 \rightarrow \neg f_2)$). If we denote the set of formulas which use the symbols $\exists, \neg, \wedge$ by $\mathrm{FOL}_{\exists,\neg,\wedge}$ and if we denote the set of formulas which use the symbols $\exists, \neg, \rightarrow$ by $\mathrm{FOL}_{\exists,\neg,\rightarrow}$, we can translate each formula $f \in \mathrm{FOL}_{\exists,\neg,\wedge}$ to a formula $f^* \in \mathrm{FOL}_{\exists,\neg,\rightarrow}$, and vice versa. Of course we can define canonically the relation $\models$ between relational structures and formulas of $\mathrm{FOL}_{\exists,\neg,\rightarrow}$ as well. It is easy to show that we have

$$\mathcal{M} \models f \quad \Longleftrightarrow \quad \mathcal{M} \models f^*$$

for all relational structures $\mathcal{M}$ and all formulas $f \in \mathrm{FOL}_{\exists,\neg,\wedge}$. So we can immediately carry over results from $\mathrm{FOL}_{\exists,\neg,\rightarrow}$ to $\mathrm{FOL}_{\exists,\neg,\wedge}$ and vice versa. In particular we will argue that the calculus we present is sound and complete in $\mathrm{FOL}_{\exists,\neg,\rightarrow}$, hence it is sound and complete in $\mathrm{FOL}_{\exists,\neg,\rightarrow}$.

In Fig. 8.1 we list all axioms and rules for the FOL-calculus we use in this treatise. With this rules, we define the relation $\vdash$ as follows:

Definition 8.8 (Proofs).

Let F be a set of formulas and let f be a formula. A sequence $(f_1, \ldots, f_n)$ is called proof for f from F *or* derivation of f from F*, if $f_n = f$ and for each $i = 1, \ldots, n$, one of the following conditions holds:*

- *$f_i \in F$, or*
- *there are f_j, f_k with $j, k < i$ and f_i is derived from f_j, f_k using* MP*, or*

Let $\alpha, \alpha_1, \alpha_2, \alpha_3, \ldots$ be variables and let f, g, h be formulas. Then we have the following axioms and rules in FOL:

MP: $f, f \to g \quad \vdash \quad g$

P1: $\vdash\ f \to (g \to f)$

P2: $\vdash\ (\neg f \to \neg g) \to (g \to f)$

P3: $\vdash\ (f \to (g \to h)) \to ((f \to g) \to (f \to h))$

Ex1: $\vdash\ f \to \exists\alpha.f$

Ex2: $\vdash\ f \to \exists\alpha_1.f[\alpha_1/\alpha_2]$ if $\alpha_1 \notin Free(f)$

Ex3: $\vdash\ f[\alpha/c] \to \exists\alpha.f$ for $c \in \mathcal{G}$

Ex4: $f \to g \vdash \exists\alpha.f \to g$ if $\alpha \notin Free(g)$

Id1: $\vdash\ \alpha_0 = \alpha_0$

Id2: $\vdash\ \alpha_0 = \alpha_1 \to \alpha_1 = \alpha_0$

Id3: $\vdash\ \alpha_0 = \alpha_1 \to \alpha_1 = \alpha_2 \to \alpha_0 = \alpha_2$

Cong1: $\vdash\ \alpha_0 = \alpha_1 \to C(\alpha_0) \to C(\alpha_1)$ for $C \in \mathcal{C}$

Cong2: $\vdash\ \alpha_0 = \alpha_n \to \alpha_1 = \alpha_{n+1} \to \ldots \to \alpha_{n-1} = \alpha_{2n-1}$
$\to R(\alpha_0, \ldots, \alpha_{n-1}) \to R(\alpha_n, \ldots, \alpha_{2n})$

Ax1: $\vdash\ \top(\alpha)$

Ax2: $\vdash\ C_1(\alpha) \to C_2(\alpha)$ for $C_1, C_2 \in \mathcal{C}$ with $C_1 \leq_{\mathcal{C}} C_2$

Ax3: $\vdash\ R_1(\alpha_1, \ldots, \alpha_n) \to R_2(\alpha_1, \ldots, \alpha_n)$
for $R_1, R_2 \in \mathcal{R}_k$ with $R_1 \leq_{\mathcal{R}} R_2$

Fig. 8.1. The FOL-calculus we use in this treatise.

- *there is* f_j *with* $j < i$ *and* f_i *is derived from* f_j *using* Ex4*, or*
- f_i *is one of the remaining axioms.*

If there is a derivation of f *from* F*, we write* $F \vdash f$*. We write* $g \vdash f$ *instead of* $\{g\} \vdash f$ *for formulas* g*. Two formulas* g *with* $f \vdash g$ *and* $g \vdash f$ *are called* provably equivalent*.*

The rules MP, P1, P2, P3 form a sound and complete calculus for propositional logic (see [69]). Formulas f which can be derived from $\emptyset$ only with these rules are called *tautologous*.

The rules Ex1, Ex2, Ex3, Ex4, Id1, Id2, Id3, Cong2 form the step from propositional logic to first order predicate logic. The rules Ex1-Ex4 are common rules which are needed when the existential quantifier is introduced (see for example [57]). The rules Id1, Id2, Id3, Cong2 are well-known rules which are used when the identity relation is used.

We mentioned in the beginning of Sect. 8.2 that the classical relational structures are pairs $\mathcal{M} := (U, I)$, consisting of a universe U and a function $I := I_{\mathcal{G}} \dot{\cup} I_{\mathcal{R}}$ with $I_{\mathcal{G}} : \mathcal{G} \to M$ and $I_{\mathcal{R}} : \mathcal{R}_k \to \mathfrak{P}(M^k)$ for each k. It is well known that the axioms and rules Ex1, Ex2, Ex3, Ex4, Id1, Id2, Id3, Cong2 are sound and complete, if we consider these structures. But we have slightly changed the classical relational structures in order to define relational structures which respect the properties of the alphabet $\mathcal{A}$. As we distinguish between concept names and unary relation names, we have two congruence axioms Cong1 and Cong2 instead of one only for relation names. Furthermore, we have a special concept name $\top$, and we have orders on the set $\mathcal{C}$ of concept names and on the set $\mathcal{R}$ of relation names. For this reason, we have added the axioms Ax1, Ax2, Ax3 to the calculus. Now we have that the relational structures we have defined are exactly the classical relational structures which fulfill the axioms Ax1, Ax2 and Ax3 (and which distinguish between concept names and names of unary relations). From this we conclude immediately the following theorem:

Theorem 8.9 (Soundness and Completeness of FOL).
A set of formulas F and a formula f over $\mathcal{A}$ satisfy

$$F \vdash f \quad \Longleftrightarrow \quad F \models f$$

9 Semantics for Existential Concept Graphs

In Chap. 4 we provided contextual structures as models for nonexistential concept graphs with cuts, i.e., we provided a contextual semantics. These semantics can be seen as a combination of the contextual approach of Prediger for concept graphs without negation, the game-theoretical semantics by Hintikka for concept graphs with negation, and the semantics of Peirce for existential graphs. In Sect. 9.1 we will extend this approach to *existential* concept graphs with cuts. We want to point out that this is a kind of semantics which is intended to be part of *'Contextual Logic'* (see [73], [75], [77], [79]).

On the other hand, as we have mentioned in Chap. 4, the most established semantics for conceptual graphs is a mapping of conceptual graphs to FOL-formulas with the operator Φ. As Φ is a mapping of one syntactically given logical language into another syntactically given logical language, in our view the use of the term 'semantics' for Φ is not appropriate. We consider Φ as a translation instead. In Sect. 9.2 we will provide a mathematical definition for $\Phi : \mathrm{CG} \to \mathrm{FOL}$ as well as a mathematical definition for a mapping $\Psi : \mathrm{FOL} \to \mathrm{CG}$ which, as we will show later, can be seen as the inverse mapping of Φ.

The semantics for FOL are usually based on relational models (see Chap. 8). So, via the translation Φ, these relational structures can also serve as models for concept graphs with cuts (i.e., although we do not consider Φ as a *direct* semantics for concept graphs, we yield an extensional semantics for concept graphs *via* Φ). So we have provided a semantics for concept graphs in two different ways: We have contextual structures which serve directly as models for concept graphs, and we have relational structures which serve indirectly via the translation Φ as models for concept graphs. But relational structures are closely related to contextual structures: It is easy to transform them into each other (see Def. 9.6 and Def. 9.7). Using this transformation, we will show that the two different semantics yield the same semantical entailment relation between concept graphs (see Thm. 9.10). This will be worked out in Sect. 9.3.

F. Dau: The Logic System of Concept Graphs with Negation, LNAI 2892, pp. 93-105, 2003.

9.1 Contexual Semantics

When an existential concept graph is evaluated in a contextual structure $(\vec{\mathbb{K}}, \lambda)$, we have to assign objects of our universe of discourse G_0 to its generic markers. This is done – analogously to FOL (see Def. 8.7) – by valuations.

Definition 9.1 (Partial and Total Valuations).
Let $\mathfrak{G} := (V, E, \nu, \top, Cut, area, \kappa, \rho)$ *be an existential concept graph with cuts and let* $(\vec{\mathbb{K}}, \lambda)$ *be a contextual structure over* $\mathcal{A}$*. A mapping* $ref : V' \to G_0$ *with* $V^{\mathcal{G}} \subseteq V' \subseteq V$ *and* $ref(v) = \lambda_{\mathcal{G}}(\rho(v))$ *for all* $v \in V^{\mathcal{G}}$ *is called a* partial valuation of $\mathfrak{G}$*. If* $V' \supseteq \{v \in V^* \mid v > c\}$ *and* $V' \cap \{v \in V^* \mid v \leq c\} = \emptyset$ *then we say that* ref *is a partial valuation* for the context c*. If* $V' = V$ *then* ref *is called* (total) valuation of $\mathfrak{G}$*.*

Now we can define whether a concept graph is valid in a contextual structure over $\mathcal{A}$. This shall be done in two ways. The first way is directly adopted from FOL (see Def. 8.7). In FOL, we start with total valuations of the variables of the formula, i.e., an object is assigned to each variable. Whenever an $\exists$-quantifier is evaluated during the evaluation of the formula, the object which is assigned to the quantified variable is substituted (i.e., the total valuation is successively changed). As we adopt this classical approach of FOL for concept graphs with cuts, we denote the semantical entailment relation by '$\models_{class}$'. It is defined as follows:

Definition 9.2 (Classical Evaluation of Graphs).
Let $\mathfrak{G} := (V, E, \nu, \top, Cut, area, \kappa, \rho)$ *be an existential concept graph with cuts and let* $(\vec{\mathbb{K}}, \lambda)$ *be a contextual structure over* $\mathcal{A}$*. Inductively over the tree* $Cut \cup \{\top\}$*, we define*[1] $(\vec{\mathbb{K}}, \lambda) \models_{class} \mathfrak{G}[c, ref]$ *for each context* $c \in Cut \cup \{\top\}$ *and every total valuation* $ref : V \to G_0$*:*

$(\vec{\mathbb{K}}, \lambda) \models_{class} \mathfrak{G}[c, ref] :\Longleftrightarrow$

it exists a valuation $\widetilde{ref} : V \to G_0$ *with* $\widetilde{ref}(v) = ref(v)$ *for all* $v \in V \backslash area(c)$ *such that the following conditions hold:*

- $\widetilde{ref}(v) \in Ext(\lambda_{\mathcal{C}}(\kappa(v)))$ *for each* $v \in V \cap area(c)$ *(vertex condition)*
- $\widetilde{ref}(e) \in Ext(\lambda_{\mathcal{R}}(\kappa(e)))$ *for each* $e \in E \cap area(c)$ *(edge condition)*
- $(\vec{\mathbb{K}}, \lambda) \not\models_{class} \mathfrak{G}[d, \widetilde{ref}]$ *for each* $d \in Cut \cap area(c)$ *(cut condition and iteration over* $Cut \cup \{\top\}$*)*

[1] In order to be in line with the notion '$\mathcal{M} \models_{val} f$' of FOL (see Def.8.7), it would be preferable to use a notion '$(\vec{\mathbb{K}}, \lambda) \models_{c,ref} \mathfrak{G}$' for concept graphs. But as we have *two different* entailment relations in CG, we decided to disginguish them by writing '$\models_{class}$' and '$\models_{endo}$' (see Def. 9.3). Thus, as the symbol '$\models$' is already indexed, we write the evaluated context c and the valuation ref in square brackets instead.

If there is a total valuation ref *such that* $(\vec{\mathbb{K}},\lambda) \models_{class} \mathfrak{G}[\top, ref]$, *we write* $(\vec{\mathbb{K}},\lambda) \models_{class} \mathfrak{G}$. *If we have two concept graphs* $\mathfrak{G}_1$, $\mathfrak{G}_2$ *such that* $(\vec{\mathbb{K}},\lambda) \models_{class} \mathfrak{G}_2$ *for each contextual structure* $(\vec{\mathbb{K}},\lambda)$ *with* $(\vec{\mathbb{K}},\lambda) \models_{class} \mathfrak{G}_1$, *we write* $\mathfrak{G}_1 \models_{class} \mathfrak{G}_2$.

The second semantics we will provide is adopted from the method of Peirce to evaluate existential graphs. He read and evaluated existential graphs from the outside, hence starting with the sheet of assertion, and proceeded with the inner cuts. During this evaluation, he assigned successively values to the lines of identity. This method (reading a graph from the outside and proceeding inwardly) is what he called 'endoporeutic method' (see [52]). We adopt this approach for concept graphs with cuts and denote the corresponding semantical entailment relation by '$\models_{endo}$'. But before we give a precise definition of '$\models_{endo}$', we exemplify it on the graph $\mathfrak{G}$ from Fig. 2.1 on p. 33.

We start the evaluation of $\mathfrak{G}$ on the sheet of assertion $\top$. As only the cut c_1 is directly enclosed by $\top$, $\mathfrak{G}$ is true if the part of $\mathfrak{G}$ which is enclosed by c_1 is false. As c_1 contains the vertex v_1 and the cut c_2, we now have the following: $\mathfrak{G}$ is true if it is not true that there exists an object o_1 such that o_1 is a cat and the proposition which is enclosed by c_2 is false. Now we have to evaluate the area of c_2. Intuitively spoken, the area of c_2 becomes true if there is an object that is a cute animal and identical to o_1. So, during this step of the evaluation, we refer to the object o_1 (that is why the endoporeutic method proceeds *inwardly*: We cannot evaluate the inner cut c_2 unless we know which object is assigned to the generic marker of v_1. Please note that the assignment we have build so far is a partial valuation for the cut c_2.) Hence $\mathfrak{G}$ is true if there is no cat such that there is no other object which is identical to the cat and which is a cute animal. In simpler words: $\mathfrak{G}$ is true if there is no cat which is not a cute animal, i.e., if every cat is a cute animal.

Definition 9.3 (Endoporeutic Evaluation of Graphs).

Let $\mathfrak{G} := (V, E, \nu, \top, Cut, area, \kappa, \rho)$ *be an existential concept graph with cuts and let* $(\vec{\mathbb{K}},\lambda)$ *be a contextual structure over* $\mathcal{A}$. *Inductively over the tree* $Cut \cup \{\top\}$, *we define* $(\vec{\mathbb{K}},\lambda) \models_{endo} \mathfrak{G}[c, ref]$ *for each context* $c \in Cut \cup \{\top\}$ *and every partial valuation* $ref : V' \subseteq V \to G_0$ *for c:*

$(\vec{\mathbb{K}},\lambda) \models_{endo} \mathfrak{G}[c, ref] :\Longleftrightarrow$

> ref *can be extended to a partial valuation* $\widetilde{ref} : V' \cup (V \cap area(c)) \to G_0$ *(i.e.,* $\widetilde{ref}(v) = ref(v)$ *for all* $v \in V'$*), such that the following conditions hold:*
>
> - $\widetilde{ref}(v) \in Ext(\lambda_{\mathcal{C}}(\kappa(v)))$ *for each* $v \in V \cap area(c)$ *(vertex condition)*
> - $\widetilde{ref}(e) \in Ext(\lambda_{\mathcal{R}}(\kappa(e)))$ *for each* $e \in E \cap area(c)$ *(edge condition)*
> - $(\vec{\mathbb{K}},\lambda) \not\models_{endo} \mathfrak{G}[d, \widetilde{ref}]$ *for each* $d \in Cut \cap area(c)$ *(cut condition and iteration over* $Cut \cup \{\top\}$*)*

For $(\vec{\mathbb{K}}, \lambda) \models_{endo} \mathfrak{G}[\top, \emptyset]$ *we write* $(\vec{\mathbb{K}}, \lambda) \models_{endo} \mathfrak{G}$*. If we have two concept graphs* $\mathfrak{G}_1$*,* $\mathfrak{G}_2$ *such that* $(\vec{\mathbb{K}}, \lambda) \models_{endo} \mathfrak{G}_2$ *for each contextual structure* $(\vec{\mathbb{K}}, \lambda)$ *with* $(\vec{\mathbb{K}}, \lambda) \models_{endo} \mathfrak{G}_1$*, we write* $\mathfrak{G}_1 \models_{endo} \mathfrak{G}_2$*.*

We want to stress that this definition (namely the edge condition) relies on the condition that we consider concept graphs with dominating nodes (this will be discussed further in Sect. 14.3).

The main difference of the last definition to Def. 9.2 is the following: In Def. 9.2, we start with a total valuation of the generic nodes which is, during the evaluation. successively changed. In the Def. 9.3, we start with the empty partial valuation which is, during the evaluation, successively completed. Unsurprisingly, these two definitions yield exactly the same entailment relations:

Lemma 9.4 (Both Evaluations are Equivalent).

Let $\mathfrak{G} := (V, E, \nu, \top, Cut, area, \kappa, \rho)$ *be an existential concept graph with cuts and let* $(\vec{\mathbb{K}}, \lambda)$ *be a contextual structure over* $\mathcal{A}$*. Then we have*

$$(\vec{\mathbb{K}}, \lambda) \models_{class} \mathfrak{G} \quad \Longleftrightarrow \quad (\vec{\mathbb{K}}, \lambda) \models_{endo} \mathfrak{G}$$

Proof: We show inductively over $Cut \cup \{\top\}$ that the following conditions are satisfied for every context $c \in Cut \cup \{\top\}$ and every partial valuation $ref' : V' \subseteq V \to G_0$ for the context c:[2]

$$\begin{aligned} &(\vec{\mathbb{K}}, \lambda) \models_{endo} \mathfrak{G}[c, ref'] && (9.1)\\ &\Longleftrightarrow (\vec{\mathbb{K}}, \lambda) \models_{class} \mathfrak{G}[c, ref] \text{ for all extensions } ref : V \to G_0 \text{ of } ref' && (9.2)\\ &\Longleftrightarrow (\vec{\mathbb{K}}, \lambda) \models_{class} \mathfrak{G}[c, ref] \text{ for one extension } ref : V \to G_0 \text{ of } ref' && (9.3)\end{aligned}$$

As $\mathfrak{G}$ has dominating nodes, we have

$$V_e \subseteq \{v \in V \mid v \geq c\} \text{ for each } e \in E \cap area(c) \qquad (9.4)$$

Now the proof is done by induction over $Cut \cup \{\top\}$. Let c be a context such that (9.1) $\Longleftrightarrow$ (9.2) $\Longleftrightarrow$ (9.3) for each cut $d < c$.

We start with the proof of (9.1) $\Longrightarrow$ (9.2), so let $ref' : V' \subseteq V \to G_0$ be a partial valuation for c such that $(\vec{\mathbb{K}}, \lambda) \models_{endo} \mathfrak{G}[c, ref]$. Hence there is a partial valuation $\widetilde{ref'}$ which extends ref' to $V' \cup (V \cap area(c))$ and which fulfills the properties in Def. 9.3. Furthermore, let $ref : V \to G_0$ be an arbitrary total valuation which extends ref' to V. We want to show $(\vec{\mathbb{K}}, \lambda) \models_{class} \mathfrak{G}[c, ref]$. We set

[2] The ongoing proof of Lem. 9.4 relies on the idea that a valuation does not determine which values are assigned to the vertices $v \in V^* \cap area(c)$. For this reason it is crucial to consider only partial valuations *for the context* c, i.e., partial valuations ref' which satisfy in particular $dom(ref') \cap \{v \in V^* \mid v \leq c\} = \emptyset$.

$$\widetilde{ref} := ref|_{V\backslash area(c)} \cup \widetilde{ref'}|_{area(c)}$$

As ref is an extension of ref', we have $\widetilde{ref}|_{\{v\in V \mid v>c\}} = \widetilde{ref'}|_{\{v\in V \mid v>c\}}$, and by definition of $\widetilde{ref}$, we have $\widetilde{ref}|_{area(c)} = \widetilde{ref'}|_{area(c)}$. From this we conclude $\widetilde{ref}|_{\{v\in V \mid v\geq c\}} = \widetilde{ref'}|_{\{v\in V \mid v\geq c\}}$. Since all vertex and edge conditions hold for $\widetilde{ref'}$, (9.4) yields that all vertex and edge conditions hold for $\widetilde{ref}$, too. Furthermore we have $(\vec{\mathbb{K}}, \lambda) \not\models_{endo} \mathfrak{G}[d, ref']$ for each $d < c$. Since $\widetilde{ref}$ is an extension of $\widetilde{ref'}$, the induction hypothesis (9.3) $\Longrightarrow$ (9.1) for d yields $(\vec{\mathbb{K}}, \lambda) \not\models_{class} \mathfrak{G}[d, ref']$. So the proof for (9.1) $\Longrightarrow$ (9.2) is done.

The implication (9.2) $\Longrightarrow$ (9.3) holds trivially.

Finally, we show that (9.3) $\Longrightarrow$ (9.1) holds for c. So assume that (9.3) is true for c, i.e., there is an extension $ref : V \to G_0$ of ref' with such that $(\vec{\mathbb{K}}, \lambda) \models_{class} \mathfrak{G}[c, ref]$. This means that there is a valuation $\widetilde{ref}$ with $\widetilde{ref}(v) = ref(v)$ for all $v \in V\backslash area(c)$ and which fulfills the properties in Def. 9.2. Now define

$$\widetilde{ref'} := ref' \cup \widetilde{ref}|_{area(c)}$$

Again, $\widetilde{ref}$ is an extension of $\widetilde{ref'}$, and $\widetilde{ref}|_{\{v\in V \mid v\geq c\}} = \widetilde{ref'}|_{\{v\in V \mid v\geq c\}}$. Similar arguments as in the proof for (9.1) $\Longrightarrow$ (9.2) yield that all edge and vertex conditions hold for $\widetilde{ref'}$. Furthermore, if $d < c$ is a cut, the cut-condition $(\vec{\mathbb{K}}, \lambda) \not\models_{endo} \mathfrak{G}[d, ref']$ holds because of the induction hypothesis (9.1) $\Longrightarrow$ (9.2). So we have that $\widetilde{ref'}$ fulfills the properties in Def. 9.3, and the proof for the implication (9.3) $\Longrightarrow$ (9.1) is done. □

As Lem. 9.4 shows that Def. 9.2 and and Def. 9.3 yield the same entailment relation between models and graphs, we will write $\models$ instead of $\models_{endo}$ and $\models_{class}$. Nevertheless it will turn out that in some proofs it is more useful to use Def. 9.2 for $\models$, and in other proofs it is more useful to use Def. 9.3.

9.2 Syntactical Translations

In this section we will provide the definitions for mappings $\Psi : \text{FOL} \to \text{CG}$ and $\Phi : \text{CG} \to \text{FOL}$ and which can be understood as translations between the two logical systems FOL and CG. According to the structures of formulas resp. graphs, these mappings are defined recursively.

We have to cope with some problems caused by structural differences between the logical systems FOL and CG. To go into details:

1. In CG, we have no syntactical devices which correspond to the free variables of FOL, but (as Ψ is defined recursively), we need translations from formulas with free variables to concept graphs. For this reason we consider concept graphs *with (free) variables* $*_\alpha$ in this section, but their

introduction should be understood as a mere technical trick. It will turn out that concept graph without variables will be translated to formulas without free variables, and vice versa.

2. In FOL, we have an infinite set of variables which are used to range over objects. In CG, only the generic marker '$*$' is used for this purpose. Thus a formula $\Phi(\mathfrak{G})$ will be only given up to the names of the variables.

   In FOL, we can syntactically express different orders of formulas in conjunctions. As conjunction is an assossiative and commutative operation, in FOL the calculus allows to change the order of formulas in conjunctions. In CG, conjunction is expressed by the juxtaposition of graphs. Thus we have no possibility to express different orderings of graphs in conjunctions. For the mapping Φ this yields the following conclusion: A formula $\Phi(\mathfrak{G})$ of a concept graph $\mathfrak{G}$ is moreover only given up the order of the subformulas of conjunctions.

Now we are prepared to provide the definitions of Ψ and Φ.

Definition of Ψ:

Before we provide a translation of formulas to concept graphs, we have to translate the terms. This is done canonically by a mapping Ψ_{term}. We set $\Psi_{term}(g) \ := \ g$ for each object name $g \in \mathcal{G}$ and we set $\Psi_{term}(\alpha) \ := \ *_\alpha$ for variables $\alpha \in$ Var. Now we can define Ψ inductively over the composition of formulas (see Def. 8.1). For each case, we first provide a diagram or an informal description before we state the explicit mathematical definition.

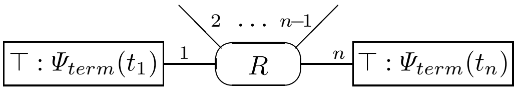
– $C(t)$ for a term t and a concept name C: $\Psi(C(t)) := \boxed{C : \Psi_{term}(t)}$

  i.e., we set $\Psi(C(t)) := \mathfrak{G}$ with

  $\mathfrak{G} \ := \ (\{1\}, \emptyset, \emptyset, \top, \emptyset, \{(\top, \{1\})\}, \{(1, C)\}, \{(1, \Psi_{term}(t))\})$

– $R(t_1, \ldots, t_n)$ for a n-ary relation name R and terms $t_1, \ldots, t_n$:

  $\Psi(R(t_1, \ldots, t_n)) \ :=$



  i.e., we set $\Psi(R(t_1, \ldots, t_n)) := \mathfrak{G}$ with

  $$\begin{aligned}\mathfrak{G} := (&\{1, \ldots, n\}, \{0\}, \{(0, (1, \ldots, n))\}, \top, \emptyset, \{(\top, \{0, 1, \ldots, n\})\},\\ &\{(0, R), (1, \top), \ldots, (n, \top)\}, \{(1, \Psi_{term}(t_1)), \ldots, (n, \Psi_{term}(t_n))\})\end{aligned}$$

– $f_1 \wedge f_2$ for two formulas f_1 and f_2: $\quad \Psi(f_1 \wedge f_2) \ := \ \Psi(f_1) \quad \Psi(f_2)$

  (i.e., $\Psi(f_1 \wedge f_2)$ is the juxtaposition of $\Psi(f_1)$ and $\Psi(f_2)$).

– $\neg f$ for a formula f: $\qquad \Psi(\neg f) \ := \ \boxed{\Psi(f)}$

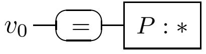
For $\Psi(f) = (V, E, \nu, \top, Cut, area, \kappa, \rho)$ let $c_0 \notin V \cup E \cup Cut \cup \{\top\}$ be a new cut. Now we set $\Psi(\neg f) := (V, E, \nu, \top, Cut', area', \kappa, \rho)$ with

$$\begin{aligned}&Cut' = Cut \cup \{c_0\} \text{ and}\\ &area' = area \backslash \{(\top, area(\top))\} \,\cup\, \{(c_0, area(\top)), (\top, c_0)\}\end{aligned}$$

- $\exists\alpha.f$ for a formula f and a variable α (this case is called *existential step*):

  If $\alpha \notin Free(f)$, we set $\Psi(\exists\alpha.f) \;:=\; \Psi(f)$.

  For $\alpha \in Free(f)$, the following steps have to be carried out:

  1. A new concept box $v_0 := \boxed{\top : *}$ is juxtaposed to $\Psi(f)$.
  2. For every edge e, every concept box $\boxed{\top : *_\alpha}$ which is incident with e is substituted by the new concept box v_0 (concept boxes which are incident with an edge can only carry the concept name $\top$).
  3. Every isolated concept box $\boxed{P : *_\alpha}$ is substituted by a concept box $\boxed{P : *}$, which is linked to the new concept box v_0 with an identity link:  .
  4. All concept boxes $\boxed{\top : *_\alpha}$ are erased (as we performed step 2, all these boxes are isolated).

  The mathematically precise procedure is as follows:

  Let $\Psi(f) := \mathfrak{G} := (V, E, \nu, \top, Cut, area, \kappa, \rho)$. Let v_0 be a new vertex with $v_0 \notin V \cup E \cup Cut \cup \{\top\}$. Let V_α^{iso} be the set of all isolated $*_\alpha$-vertices, i.e., $V_\alpha^{iso} := \{w \in V \mid \rho(w) = *_\alpha \,\wedge\, E_w = \emptyset\}$ and let V_α^{edge} be the set of all $*_\alpha$-vertices which are incident with an edge, i.e., we define $V_\alpha^{edge} := \{w \in V \mid \rho(w) = *_\alpha \,\wedge\, E_w \neq \emptyset\}$ (we have $\kappa(w) = \top$ for each vertex $w \in V_\alpha^{edge}$). Assign to each $w \in V_\alpha^{iso}$ a new edge e_w such that $V \cup \{v_0\}$, E, $\{e_w \,|\, w \in V_\alpha^{iso}\}$, and $Cut \cup \{\top\}$ are pairwise disjoint sets. Now we can define $\Psi(\exists x.f) := (V', E', \nu', \top', Cut', area', \kappa', \rho')$ as follows:

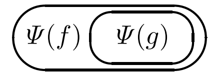
$$\begin{aligned}
V' &= (V \backslash V_\alpha^{edge}) \cup \{v_0\}\\
E' &= E \cup \{e_w \,|\, w \in V_\alpha^{iso}\}\\
\nu' &: \text{ For } e \in E \text{ we set } \nu'(e)\big|_i := \begin{cases} \nu(e)\big|_i & \text{for } \rho(e\big|_i) \neq *_\alpha\\ v_0 & \text{for } \rho(e\big|_i) = *_\alpha\end{cases}\\
&\quad\;\, \text{For } w \in V_\alpha^{iso} \text{ we set } \nu'(e_w) := (v_0, w)\\
\top' &= \top\\
Cut' &= Cut\\
area'(c) &= (area(c) \backslash V_\alpha^{edge}) \cup \{e_w \,|\, w \in V_\alpha^{iso} \cap area(c)\}\\
&\quad\;\, \text{for } c \neq \top\\
area'(\top) &= (area(\top) \backslash V_\alpha^{edge}) \cup \{e_w \,|\, w \in V_\alpha^{iso} \cap area(\top)\} \cup \{v_0\}\\
\kappa' &= \kappa|_{V \backslash V_\alpha^{edge} \cup E} \cup \{(v_0, \top)\} \cup \{(e_w, \doteq) \,|\, w \in V_\alpha^{iso}\}\\
\rho' &: \; \rho'(v) := \begin{cases} * & \text{for } v = v_0 \text{ or } v \in V \backslash V_\alpha^{edge} \text{ with } \rho(v) = *_\alpha\\ \rho(v) & \text{otherwise}\end{cases}
\end{aligned}$$

This completes the definition of Ψ. In particular, Ψ translates formulas without free variables to simple concept graphs with negations and dominating nodes.

In Chap. 2 we had shown that the ordered set of contexts $(Cut \cup \{\top\}, \leq)$ is a tree (see Cor. 2.5) and can be considered to be the 'skeleton' of a concept graph with cuts. In Chap. 8, Def. 8.5 we had defined a corresponding structure for FOL-formulas. Now the inductive definition of Ψ yields the following: If f is a formula and $\Psi(f) := \mathfrak{G} := (V, E, \nu, \top, Cut, area, \kappa, \rho)$ then it is evident that $(Neg_f, \leq)$ and $(Cut \cup \{\top\}, \leq)$ are isomorphic orders. We denote the canonically given isomorphism by $\Psi_{Neg} : Neg_f \to Cut \cup \{\top\}$.

As implications are important (e.g. nearly all axioms and rules of the FOL-calculus are build up from implications), we want to remark the following: If f and g are sentences, then we have

$$\Psi(f \to g) = \Psi(\neg(f \wedge \neg g)) =$$



This device of two nested cuts is what Peirce called a *scroll*. Scrolls are the kind how implications are written down in existential graphs and in concept graphs.

As we have finished the definition of Ψ : FOL $\to$ CG, we proceed with the definition of Φ : CG $\to$ FOL.

Definition of Φ.

Let $\mathfrak{G} := (V, E, \nu, \top, Cut, area, \kappa, \rho)$ be a simple concept graph with cuts, variables, and dominating nodes. We set $Free(\mathfrak{G}) := \{\alpha \in \mathrm{Var} \,|\, \exists v \in V.\, \rho(v) = *_\alpha\}$, and we assign to each vertex $v \in V^*$ a fresh variable $\alpha_v \notin Free(\mathfrak{G})$, so that we can define the following mapping Φ_t on V:

$$\Phi_t(v) := \begin{cases} \alpha_v \text{ for } \rho(v) = * \\ \alpha \text{ for } \rho(v) = *_\alpha \text{ and } \alpha \in \mathrm{Var} \\ g \text{ for } \rho(v) = g \text{ and } g \in \mathcal{G} \end{cases}$$

Let $\alpha_{empty} \notin Free(\mathfrak{G}) \cup \{\alpha_v \,|\, v \in V \text{ and } \rho(v) = *\}$ be a further variable. Now, inductively over the tree $Cut \cup \{\top\}$, we assign to each context $c \in Cut \cup \{\top\}$ a formula $\Phi(\mathfrak{G}, c)$. So let c be a context such that $\Phi(\mathfrak{G}, d)$ is already defined for each cut $d < c$. First, we define a formula f which encodes all edges and vertices which are directly enclosed by c. Hence, if c does not directly enclose any edges or vertices, simply set $f := (\exists \alpha_{empty}.\top(\alpha_{empty}))$. Otherwise, let f be the conjunction of the following atomic formulas:[3]

[3] Like in Def. 8.1, the signs which have to be understood literally are underlined. For example, the first formula is the sequence of signs which consists of the result of the evaluation of $\kappa(w)$, a left bracket, the result of the evaluation of $\Phi_t(w)$ and a right bracket.

To illustrate the mappings Φ and Ψ, we give a small sample. A well-known example for translating first order logic to existential graphs is the formula which expresses that a binary relation F is a (total) function. This formula is written down only by using the $\exists$-quantifier and the junctors $\wedge$ and $\neg$. It is

$$f \;:=\; \neg\exists x.\neg\exists y.(xFy \wedge \neg\exists z.(xFz \wedge \neg(y = z)))$$

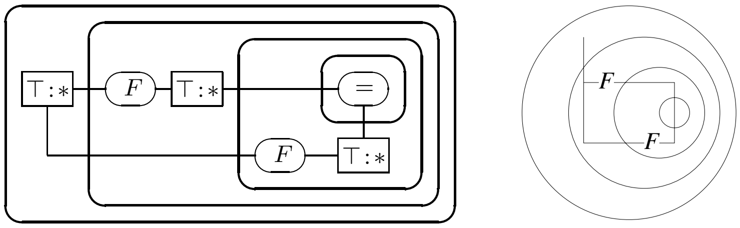


Fig. 9.1. the concept graph $\Psi(f)$ and the appropriate existential graph for f

The translations of this formula into a concept graph (by Φ) and into an existential graph are shown in Fig. 9.1. Please note the structural similarities between these two different kind of graphs. Now the concept graph in Fig. 9.1 can be translated back to a first order logic formula by the mapping Φ. One possible result (up to the chosen variables and to the order of the subformulas) is:

$$\Phi(\Psi(f)) \;=\; \neg\exists x.(\top(x) \wedge \neg\exists y.(\top(y) \wedge xFy \wedge \neg\exists.z(\top(z) \wedge xFz \wedge \neg(\wedge y = z))))$$

If we erase all subformulas $\top(\ldots)$ of this formula, we gain f again. In particular, we have $f \vdash \Phi(\Psi(f))$ and $\Phi(\Psi(f)) \vdash f$ (see Thm. 11.8).

9.3 Semantical Equivalence

Contextual structures can be considered to be an extension of relational structures. Roughly spoken: If we remove all the intensional information from a contextual structure, we get a relational structure. This yields the following definition:

Definition 9.6 (Rel. Structure Induced by Context. Structure).

Let $(\vec{\mathbb{K}}, \lambda)$ *be a contextual structure. Then let* $\mathcal{M}_{(\vec{\mathbb{K}},\lambda)} := (U, I)$ *be the following relational structure:* $U := G_0$*,* $I_{\mathcal{G}}(g) := \lambda_{\mathcal{G}}(g)$ *for all* $g \in \mathcal{G}$*,* $I_{\mathcal{C}}(C) := Ext(\lambda_{\mathcal{C}}(C))$ *for all* $C \in \mathcal{C}$ *and* $I_{\mathcal{R}}(R) := Ext(\lambda_{\mathcal{R}}(R))$ *for all* $R \in \mathcal{R}$*. The relational structure* $\mathcal{M}_{(\vec{\mathbb{K}},\lambda)}$ *is called the* relational structure of $(\vec{\mathbb{K}}, \lambda)$.

$$\kappa(w)\underline{(}\varPhi_t(w)\underline{)} \text{ with } w \in V \cap area(c),$$

$$\varPhi_t(w_1)\underline{=}\varPhi_t(w_2) \text{ with } k \in E^{id} \cap area(c) \text{ und } \nu(k) = (w_1, w_2), \quad \text{and}$$

$$\kappa(e)\underline{(}\varPhi_t(w_1)\underline{,}\ldots\underline{,}\varPhi_t(w_j)\underline{)} \text{ with } e \in E^{nonid} \cap area(c) \text{ and } \nu(e) = (w_1, \ldots, w_j).$$

Let $v_1, \ldots, v_n$ be the vertices of $\mathfrak{G}$ which are enclosed by c and which fulfill $\rho(v_i) = *$, and let $area(c) \cap Cut = \{c_1, \ldots, c_l\}$ (by induction, we already assigned formulas to these cuts). If $l = 0$, set $\varPhi(\mathfrak{G}, c) := \underline{\exists}\alpha_{v_1}\underline{.}\ldots\underline{\exists}\alpha_{v_n}\underline{.}f$, otherwise set

$$\varPhi(\mathfrak{G}, c) := \underline{\exists}\alpha_{v_1}\underline{.}\ldots\underline{\exists}\alpha_{v_n}\underline{.}\underline{(}f\underline{\wedge}\underline{\neg}\varPhi(\mathfrak{G}, c_1)\underline{\wedge}\ldots\underline{\wedge}\underline{\neg}\varPhi(\mathfrak{G}, c_l)\underline{)} \quad ,$$

Finally set $\varPhi(\mathfrak{G}) := \varPhi(\mathfrak{G}, \top)$, and the definition of $\varPhi$ is finished.

Let $\mathfrak{G}$ be a concept graph and $f := \varPhi(\mathfrak{G})$. Similar as for $\varPsi$, it is evident that $(Cut \cup \{\top\}, \leq)$ and $(Neg_f, \leq)$ are isomorphic quasiorders. We denote the canonically given isomorphism by $\varPhi_{Cut} : Cut \cup \{\top\} \to Neg_f$.

We want to point out that $\varPhi$ is, strictly speaking, not a function. We have assigned arbitrary variables to the generic nodes, and the order of quantifiers or formulas in conjunctions is arbitrary as well. So $\varPhi$ determines a formula only up to the names of the variables, the order of quantifiers and the order of the subformulas of conjunctions. To put it more formally: The image $\varPhi(\mathfrak{G})$ of a concept graph $\mathfrak{G}$ is only uniquely given up to the following equivalence relation:[4]

Definition 9.5 (First Equivalence Relation for Formulas). *Let $\cong$ be the smallest equivalence relation on FOL such that the following conditions hold:*

1. *If f_1, f_2, f_3 are formulas then we have $f_1 \wedge f_2 \cong f_2 \wedge f_1$ and $(f_1 \wedge f_2) \wedge f_3 \cong f_1 \wedge (f_2 \wedge f_3)$,*
2. *if f_1, f_2 are formulas with $f_1 \cong f_2$ and if α, β are variables then $\exists\alpha.\exists\beta.f_1 \cong \exists\beta.\exists\alpha.f_2$,*
3. *if f and f' are equal up to renaming bound variables then $f \cong f'$, and*
4. *if f_1, f_2, g_1, g_2 are formulas with $f_1 \cong f_2$ and $g_1 \cong g_2$ then $\neg f_1 \cong \neg f_2$, $f_1 \wedge g_1 \cong f_2 \wedge g_2$ and $\exists\alpha.f_1 \cong \exists\alpha.f_2$.*

It is well known that $\vdash f \leftrightarrow g$ for formulas f, g with $f \cong g$. In particular we have $f \vdash g$ and $g \vdash f$. So all possible images $\varPhi(\mathfrak{G})$ of a concept graph $\mathfrak{G}$ are provably equivalent and can therefore be identified, and we consider $\varPhi$ a mapping which assigns a formula to each concept graph.

[4] We provide a definition for this equivalence relation for two reasons: First, the definition explicates the explanation we have just given. Second, we will extend this definition in Sect. 11.1 where we will show that f and $\varPhi(\varPsi(f))$ are equivalent.

On the other hand, we can assign to each relational structure a power context family $\vec{\mathbb{K}}$ with adequate functions $\lambda_{\mathcal{G}}$, $\lambda_{\mathcal{C}}$ and $\lambda_{\mathcal{R}}$. This is done now:

Definition 9.7 (Context. Structure Induced by Rel. Structure).
Let $\mathcal{M} := (U, I)$ be a relational structure over $\mathcal{A}$. Then define a contextual structure $(\vec{\mathbb{K}}, \lambda)_{\mathcal{M}} := (\vec{\mathbb{K}}, \lambda)$ as follows: First define the power context family $\vec{\mathbb{K}}$:

1. *$G_0 := U$ and $G_k := U^k$ for each $k = 1, \ldots, n$,*
2. *$M_0 := \mathcal{C}$ and $M_k := \mathcal{R}_k$ for each $k = 1, \ldots, n$,*
3. *$(g, C) \in I_0 :\Longleftrightarrow g \in I(C)$ for $C \in \mathcal{C}$ and*
4. *$((g_1, \ldots, g_k), R) \in I_k :\Longleftrightarrow (g_1, \ldots, g_k) \in I(R)$ for $R \in \mathcal{R}_k$.*

*Now set $\lambda_{\mathcal{G}}(g) := I(g)$ for each $g \in \mathcal{G}$, $\lambda_{\mathcal{C}}(C) := (C^{I_0}, C^{I_0 I_0})$ and $\lambda_{\mathcal{R}}(R) := (R^{I_k}, R^{I_k I_k})$ for each $R \in \mathcal{R}_k$. Then $(\vec{\mathbb{K}}, \lambda)_{\mathcal{M}}$ is called the $\vec{\mathbb{K}}$-*interpretation *of $\mathcal{M}$.*

The mappings between CG and contextual models on the one side and FOL and relational models on the other side can now be sketched as in Figure 9.2:

$$\begin{tikzcd}
(\vec{\mathbb{K}}, \lambda) \arrow[d, shift right, "(\vec{\mathbb{K}}, \lambda) \to \mathcal{M}_{(\vec{\mathbb{K}}, \lambda)}"'] & \models & \mathfrak{G} \in \mathrm{CG} \arrow[d, shift right, "\Phi"'] \\
\mathcal{M} := (U, I) \arrow[u, shift right, "\mathcal{M} \to (\vec{\mathbb{K}}, \lambda)_{\mathcal{M}}"'] & \models & f \in \mathrm{FOL} \arrow[u, shift right, "\Psi"']
\end{tikzcd}$$

Fig. 9.2. the mappings between CG and FOL

The following lemma is easy to see:

Lemma 9.8.
For each relational structure $\mathcal{M}$ we have $\mathcal{M}_{(\vec{\mathbb{K}}, \lambda)_{\mathcal{M}}} = \mathcal{M}$.

Without proof.

We want to point out that the 'inverse direction' of this lemma does not hold, i.e., we do not have $(\vec{\mathbb{K}}, \lambda)_{\mathcal{M}_{(\vec{\mathbb{K}}, \lambda)}} = (\vec{\mathbb{K}}, \lambda)$. The main reason for this is as follows: Each concept name and each relation name of $\mathcal{A}$ is mapped to a concept of $\vec{\mathbb{K}}$, but $\vec{\mathbb{K}}$ may contain further concepts. When we construct $\mathcal{M}_{(\vec{\mathbb{K}}, \lambda)}$, these additional concepts are not considered, i.e., they get lost. Thus we cannot even expect that the concept lattices of $(\vec{\mathbb{K}}, \lambda)_{\mathcal{M}_{(\vec{\mathbb{K}},\lambda)}}$ and $(\vec{\mathbb{K}}, \lambda)$ are isomorphic. Due to this point, contextual structures are richer than relational structures. This is mainly caused by the 'intensional information' which is encoded only in contextual structures.

If we now look back into the definition of the relation $\models$ between contextual structures and concept graphs, more precisely: into the vertex conditions and edge conditions of Def. 9.3 and Def. 9.2, we realize that only the extensions of formal concepts were checked. This yields the following lemma:

Lemma 9.9 (Equivalence for Rel. and Contextual Structures).

Let $\mathfrak{G} := (V, E, \nu, \top, Cut, area, \kappa, \rho)$ *be an existential concept graph with cuts and let* $(\vec{\mathbb{K}}, \lambda)$ *be a contextual structure over* $\mathcal{A}$*. Then we have the following equivalence:*

$$(\vec{\mathbb{K}}, \lambda) \models \mathfrak{G} \quad \Longleftrightarrow \quad \mathcal{M}_{(\vec{\mathbb{K}},\lambda)} \models \Phi(\mathfrak{G})$$

Proof: Let $\mathfrak{G} := (V, E, \nu, \top, Cut, area, \kappa, \rho)$. Remember that Φ is defined inductively over $Cut \cup \{\top\}$. The same holds for the relation $\models_{class}$. In the definition of $\models_{class}$, we needed total valuations $ref : V \to G_0$. In the definition of Φ, we assigned to each generic vertex $v \in V^*$ a variable $\Phi_t(v) \in$ Var. Using Φ_t, we can canonically transform each valuation $ref : V \to G_0$ to a valuation $val_{ref} : Var \to G_0$ on the set of variables by setting $val_{ref}(\alpha) := ref(\Phi_t^{-1}(\alpha))$ for each variable $\alpha \neq \alpha_{empty}$ which occurs in $\Phi(\mathfrak{G})$ (for all other variables α, the image $val_{ref}(\alpha)$ is arbitrary). Using this definition, it is easy to show inductively over $Cut \cup \{\top\}$ that, for every total valuation $ref : V \to G_0$ and every context $c \in Cut \cup \{\top\}$, we have the following equivalence:

$$(\vec{\mathbb{K}}, \lambda) \models_{class} \mathfrak{G}[c, ref] \Longleftrightarrow \mathcal{M}_{(\vec{\mathbb{K}},\lambda)} \models_{val_{ref}\Phi(\mathfrak{G},c)}$$

If we take now an arbitrary valuation ref, we have

$$\begin{aligned}(\vec{\mathbb{K}}, \lambda) \models_{class} \mathfrak{G} &\Longleftrightarrow (\vec{\mathbb{K}}, \lambda) \models_{class} \mathfrak{G}[\top, ref] \\ &\Longleftrightarrow \mathcal{M}_{(\vec{\mathbb{K}},\lambda)} \models_{val_{ref}} \Phi(\mathfrak{G}, \top) \\ &\Longleftrightarrow \mathcal{M}_{(\vec{\mathbb{K}},\lambda)} \models \Phi(\mathfrak{G})\end{aligned}$$

The last equivalence holds because $\Phi(\mathfrak{G}) = \Phi(\mathfrak{G}, \top)$ has no free variables. □

Theorem 9.10 (Main Semantical Theorem for Φ).

Let $\mathfrak{G}_1$ *and* $\mathfrak{G}_2$ *be concept graphs over* $\mathcal{A}$*. Then we have:*

$$\mathfrak{G}_1 \models \mathfrak{G}_2 \quad \Longleftrightarrow \quad \Phi(\mathfrak{G}_1) \models \Phi(\mathfrak{G}_2)$$

Proof: Assume we have $\mathfrak{G}_1 \models \mathfrak{G}_2$. Let $\mathcal{M}$ be a relational structure such that $\mathcal{M} \models \Phi(\mathfrak{G}_1)$. Then we have

$$\begin{aligned}
\mathcal{M} \models \Phi(\mathfrak{G}_1) &\overset{\text{Lem. 9.8}}{\Longleftrightarrow} \mathcal{M}_{(\vec{\mathbb{K}},\lambda)_{\mathcal{M}}} \models \Phi(\mathfrak{G}_1)\\
&\overset{\text{Lem. 9.9}}{\Longleftrightarrow} (\vec{\mathbb{K}},\lambda)(\mathcal{M}) \models \mathfrak{G}_1\\
&\overset{\text{presupposition}}{\Longrightarrow} (\vec{\mathbb{K}},\lambda)(\mathcal{M}) \models \mathfrak{G}_2\\
&\overset{\text{Lem. 9.9}}{\Longleftrightarrow} \mathcal{M}_{(\vec{\mathbb{K}},\lambda)_{\mathcal{M}}} \models \Phi(\mathfrak{G}_2)\\
&\overset{\text{Lem. 9.8}}{\Longleftrightarrow} \mathcal{M} \models \Phi(\mathfrak{G}_2)
\end{aligned}$$

Hence, one direction is shown. Now assume $\Phi(\mathfrak{G}_1) \models \Phi(\mathfrak{G}_2)$ and let $(\vec{\mathbb{K}},\lambda)$ be a contextual structure of $\mathfrak{G}_1$ (i.e., $(\vec{\mathbb{K}},\lambda) \models \mathfrak{G}_1$). Then we have

$$\begin{aligned}
(\vec{\mathbb{K}},\lambda) \models \mathfrak{G}_1 &\overset{\text{Lem. 9.9}}{\Longleftrightarrow} \mathcal{M}_{(\vec{\mathbb{K}},\lambda)} \models \Phi(\mathfrak{G}_1)\\
&\overset{\text{presupposition}}{\Longrightarrow} \mathcal{M}_{(\vec{\mathbb{K}},\lambda)} \models \Phi(\mathfrak{G}_2)\\
&\overset{\text{Lem. 9.9}}{\Longleftrightarrow} (\vec{\mathbb{K}},\lambda) \models \mathfrak{G}_2
\end{aligned}$$

and the proof is finished. □

10 Calculus for Existential Concept Graphs

In this chapter, we will provide the calculus for existential concept graphs with cuts and a couple of derived rules. Furthermore we will show that the calculus is sound with respect to the semantics we presented in the last chapter.

10.1 Calculus

The calculus for existential concept graphs which we will provide in this section is an extension of the calculus for *nonexistential* concept graphs with cuts. It consists of the same set of rules as the calculus for nonexistential concept graphs, but some rules are modified (in most cases slightly extended) in order to respect the properties of the generic marker. Similar to Sec. 5.1, we will first describe the whole calculus using common spoken language. After this, we present mathematical definitions for those rules which have a different mathematization than their nonexistential counterparts.

Definition 10.1 (Calculus for Existential Concept Graphs). *The calculus for existential concept graphs with cuts over the alphabet $\mathcal{A} := (\mathcal{G}, \mathcal{C}, \mathcal{R})$ consists of the following rules:*

- **erasure**

  In positive contexts, any directly enclosed edge, isolated vertex, and closed subgraph may be erased.
- **insertion**

  In negative contexts, any directly enclosed edge, isolated vertex, and closed subgraph may be inserted.
- **iteration**

  Let $\mathfrak{G}_0 := (V_0, E_0, \nu_0, \top_0, Cut_0, area_0, \kappa_0, \rho_0)$ be a (not necessarily closed) subgraph of $\mathfrak{G}$ and let $c \leq ctx(\mathfrak{G}_0)$ be a context such that $c \notin Cut_0$. Then a copy of $\mathfrak{G}_0$ may be inserted into c. For every vertex $v \in V_0^*$ with $ctx(v) = ctx(\mathfrak{G}_0)$, an identity-link from v to its copy may be inserted.

F. Dau: The Logic System of Concept Graphs with Negation, LNAI 2892, pp. 107-123, 2003.

- **deiteration**

  If $\mathfrak{G}_0$ *is a subgraph of* $\mathfrak{G}$ *which could have been inserted by rule of iteration, then it may be erased.*

- **double cuts**

  Double cuts (two cuts c_1, c_2 *with* $area(c_1) = \{c_2\}$*) may be inserted or erased.*

- **generalization**

  For evenly enclosed vertices and edges, their concept names or object names resp. their relation names may be generalized.

- **specialization**

  For oddly enclosed vertices and edges, their concept names or object names resp. their relation names may be specialized.

- **isomorphism**

  A graph may be substituted by an isomorphic copy of itself.

- **exchanging references**

  Let $e \in E^{id}$ *be an identity link with* $\rho(e\big|_1) = g_1$*,* $\rho(e\big|_2) = g_2$*,* $g_1, g_2 \in \mathcal{G} \cup \{*\}$ *and* $ctx(e) = ctx(e\big|_1) = ctx(e\big|_2)$*. Then the references of* v_1 *and* v_2 *may be exchanged, i.e., the following may be done: We can set* $\rho(e\big|_1) = g_2$ *and* $\rho(e\big|_2) = g_1$*.*

- **merging two vertices**

  Let $e \in E^{id}$ *be an identity link with* $\nu(e) = (v_1, v_2)$ *such that* $ctx(v_1) \geq ctx(e) = ctx(v_2)$*,* $\rho(v_1) = \rho(v_2)$ *and* $\kappa(v_2) = \top$ *hold. Then* v_1 *may be merged into* v_2*, i.e.,* v_1 *and* e *are erased and, for every edge* $e \in E$*,* $e\big|_i = v_1$ *is replaced by* $e\big|_i = v_2$*.*

- **splitting a vertex**

  Let $g \in \mathcal{G} \cup \{*\}$*. Let* $v = \boxed{P : g}$ *be a vertex in the context* c_0 *and incident with relation edges* $R_1, \ldots, R_n$*, placed in contexts* $c_1, \ldots, c_n$*, respectively. Let* c *be a context such that* $c_1, \ldots, c_n \leq c \leq c_0$*. Then the following may be done: In* c*, a new vertex* $v' = \boxed{\top : g}$ *and a new identity-link between* v *and* v' *is inserted. On* $R_1, \ldots, R_n$*, arbitrary occurences of* v *are substituted by* v'*.*

- **$\top$-erasure**

  For $g \in \mathcal{G} \cup \{*\}$*, an isolated vertex* $\boxed{\top : g}$ *may be erased from arbitrary contexts.*

- **$\top$-insertion**

  For $g \in \mathcal{G} \cup \{*\}$*, an isolated vertex* $\boxed{\top : g}$ *may be inserted in arbitrary contexts.*

- **identity-erasure**

  Let $g \in \mathcal{G}$, let $v_1 = \boxed{P_1 : g}$ and $v_2 = \boxed{P_2 : g}$ be two vertices. Then any identity-link between v_1 and v_2 may be erased.

- **identity-insertion**

  Let $g \in \mathcal{G}$, let $v_1 = \boxed{P_1 : g}$, $v_2 = \boxed{P_2 : g}$ be two vertices in contexts c_1, c_2, resp. and let $c \leq c_1, c_2$ be a context. Then an identity-link between v_1 and v_2 may be inserted into c.

The main changes between the calculus for nonexistential concept graphs and this calculus have been made in the rules 'iteration'/'deiteration' and 'splitting a vertex'/'merging two vertices'.

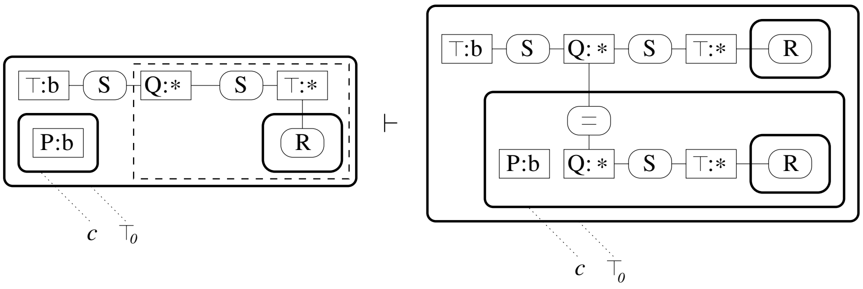
What is new in the iteration-rule is that we allowed to draw identity links from some of the vertices of the iterated subgraph to their copies. As the deiteration-rule is simply the inversion of the iteration-rule, the deiteration-rule is affected, too.

What is new in the rule 'splitting a vertex' is the following: When we split a vertex into two vertices, an identity-link has to be drawn between these two vertices. This has been added to the rule since we are allowed to split generic vertices (i.e., vertices $v \in V^*$), and without the newly added identity-link it is not clear that the two vertices which result from the splitting-operation have to refer to the same object. As the rule 'merging two vertices' is the inversion of the rule 'splitting a vertex', this rule is affected, too.

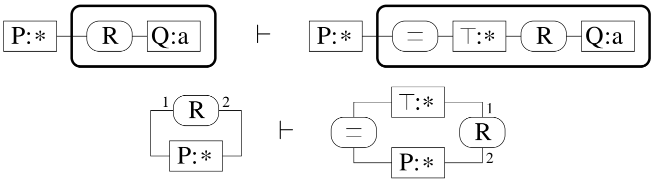
As the rules 'iteration'/'deiteration' and 'splitting a vertex'/'merging two vertices' have significantly changed, we will provide the mathematical definitions for the Beta-versions of these rules. All remaining rules have the same mathematical formalization as their nonexistential counterparts. The only difference between the rules in Alpha and Beta is: In Beta we have to consider generic markers, too. To give an examples for this: As we have $* > g$ for each $g \in \mathcal{G}$, the generalization rule now allows for positive enclosed vertices to change the reference g of the vertex to the generic marker $*$.

Before we provide the mathematical definitions for the Beta-version of the rules 'iteration'/'deiteration' and 'splitting a vertex'/'merging two vertices', we give some examples for them.

Example 10.2. Here is an example for the iteration-rule. For one of the two possible vertices, an identity link is inserted to its copy. Compare this example with the example for the iteration-rule on p. 52 in the part Alpha of this treatise.



Here are two examples for the rule ‘splitting a vertex’:



Now we are prepared to give the mathematical definition for the rules ‘iteration’/‘deiteration’ and ‘merging two vertices’/‘splitting a vertex’.

– **iteration and deiteration**

Let $\mathfrak{G} := (V, E, \nu, \top, Cut, area, \kappa, \rho)$ be a concept graph with the subgraph $\mathfrak{G}_0 := (V_0, E_0, \nu_0, \top_0, Cut_0, area_0, \kappa_0, \rho_0)$ and $c \notin Cut_0$ be context. Let $\bar{V} \subseteq \{v \in V_0 \mid ctx(v) = \top_0\}$. For each vertex $v \in \bar{V}$ let e_v be a new edge such that V, E, $\bar{E} := \{e_v \mid v \in \bar{V}\}$ and $Cut \cup \{\top\}$ are pairwise disjoint.

Now let $\mathfrak{G}' := (V', E', \nu', \top', Cut', area', \kappa', \rho')$ be the following graph:

– $V' := V \times \{1\} \,\cup\, V_0 \times \{2\}$

– $E' := E \times \{1\} \,\cup\, E_0 \times \{2\} \,\cup\, \bar{E} \times \{2\}$

– $\nu'((e,i)) := \begin{cases} ((v_1,i),\dots,(v_n,i)) \text{ for } (e,i) \in E' \backslash (\bar{E} \times \{2\}) \text{ and} \\ \qquad\qquad\qquad \nu(e) = (v_1,\dots,v_n) \\ \quad ((v,1),(v,2)) \text{ for } e = e_v \in \bar{E} \end{cases}$

– $\top' := \top$

– $Cut' := Cut \times \{1\} \,\cup\, Cut_0 \times \{2\}$

– $area'$ is defined as follows:

for $(d,i) \in Cut'$ and $d \neq c$, let $area'((d,i)) := area(d) \times \{i\}$ and

$area'((c,1)) := area(c) \times \{1\} \,\cup\, area_0(\top_0) \times \{2\} \,\cup\, \bar{E} \times \{2\}$

– $\kappa'((k,i)) := \begin{cases} \kappa(k) \text{ for } (k,i) \in V' \cup E' \backslash (\bar{E} \times \{2\}) \\ \quad \doteq \text{ for } (k,i) \in \bar{E} \times \{2\} \end{cases}$

– $\rho'((v,i)) = \rho(v)$ for all $(v,i) \in V'$

Then we say that $\mathfrak{G}'$ is derived from $\mathfrak{G}$ by *iterating the subgraph* $\mathfrak{G}_0$ *into the context* c and $\mathfrak{G}$ is derived from $\mathfrak{G}'$ by *deiterating the subgraph* $\mathfrak{G}_0$ *from the context* c.

– **merging two vertices and splitting a vertex**

Let $\mathfrak{G} := (V, E, \nu, \top, Cut, area, \kappa, \rho)$ be a concept graph. Let $e \in E^{id}$ be an identity link with $\nu(e) = (v_1, v_2)$ or $\nu(e) = (v_2, v_1)$ such that $ctx(v_1) \geq ctx(e) = ctx(v_2)$, $\rho(v_1) = \rho(v_2)$ and $\kappa(v_2) = \top$.

Now let $\mathfrak{G}' := (V', E', \nu', \top', Cut', area', \kappa', \rho')$ be the following graph:

- $V' := V \backslash \{v_2\}$
- $E' := E \backslash \{e\}$
- ν' is defined as follows: For $\nu(f) = (w_1, \ldots, w_n)$, $f \in E'$ let
$$\nu'(f) := (w'_1, \ldots, w'_n) \text{ with } w'_i := \begin{cases} w_i & \text{for } w_i \neq v_2 \\ v_1 & \text{for } w_i = v_2 \end{cases}$$
- $\top' := \top$
- $Cut' := Cut$
- $area'(c) := area(c)|_{V' \cup E' \cup Cut' \cup \{\top'\}}$ for $c \in Cut' \cup \{\top'\}$.
- $\kappa' := \kappa|_{V' \cup E'}$
- $\rho' := \rho|_{V'}$

Then we say that $\mathfrak{G}'$ is derived from $\mathfrak{G}$ by *merging* v_1 *into* v_2 and $\mathfrak{G}$ is derived from $\mathfrak{G}'$ by *splitting* v_1.

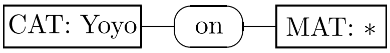
10.2 Derived Rules

In [14] we presented a preliminary version of the calculus we provide in this thesis. This calculus of [14] contains a rule, namely the 'congruence rule', which is practical to have. The congruence rule is not contained by the calculus we provide in this treatise, but it can now be derived.

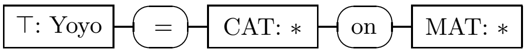
The idea of the rule is the following: Let $\mathfrak{G} := (V, E, \nu, \top, Cut, area, \kappa, \rho)$ be a concept graph, and let $v = \boxed{P : g}$ with $g \in \mathcal{G}$ be a vertex in a context c. The concept box $v =$ contains two names, namely the object name g and the concept name P. We can split v into two concept boxes such that one concept box contains the object name and the other one contains the concept name, i.e., the following may be done: In c, a new vertex $v' = \boxed{\top : g}$ and a new identity link between v and v' is inserted. Then $v = \boxed{P : g}$ is substituted by $\boxed{P : *}$.

To illustrate this procedure, we provide the example below.

We start with the graph



If we split concept box $\boxed{\text{CAT: Yoyo}}$, we get



The mathematical definition for the procedure is as follows:

- **congruence rule**

  Let $\mathfrak{G}$ be the concept graph and let $v \in V$ be a concept box with $\kappa(v) = P$, $\rho(v) = g$. Let v' be a new vertex and e' be a new edge (i.e., $V, \{v'\}$, E, $\{e'\}$, and $Cut \cup \{\top\}$ are pairwise disjoint).

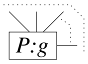
  Now let $\mathfrak{G}' := (V', E', \nu', \top', Cut', area', \kappa', \rho')$ be the following graph:
  - $V' := V \cup \{v'\}$
  - $E' := E \cup \{e'\}$
  - $\nu'(f) := \begin{cases} \nu(f) & \text{for } f \neq e' \\ (v, v') & \text{for } f = e' \end{cases}$
  - $\top' := \top$
  - $Cut' := Cut$
  - $area'(d) := \begin{cases} area(d) & \text{for } d \neq c \\ area(c) \cup \{v', e'\} & \text{for } d = c \end{cases}$
  - $\kappa'(k) := \begin{cases} \kappa(k) & \text{for } k \neq v', e' \\ \top & \text{for } k = v' \\ \doteq & \text{for } k = e' \end{cases}$
  - $\rho'(w) := \begin{cases} \rho(w) & \text{for } w \neq v, v' \\ * & \text{for } w = v \\ g & \text{for } w = v' \end{cases}$

  Then we say that $\mathfrak{G}'$ is constructed from $\mathfrak{G}$ by applying *the congruence-rule.*

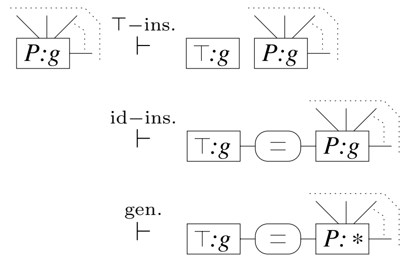
Lemma 10.3 (Congruence Rule).

Let $\mathfrak{G}'$ be constructed from $\mathfrak{G}$ by applying the congruence-rule. Then $\mathfrak{G}$ and $\mathfrak{G}'$ are provably equivalent, i.e., we have $\mathfrak{G} \vdash \mathfrak{G}'$ and $\mathfrak{G}' \vdash \mathfrak{G}$.

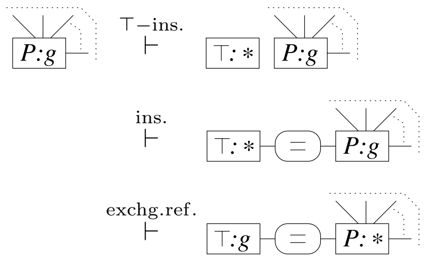
Proof: Let $v = \boxed{P : g}$ be a vertex. The fact that v is allowed to be incident with arbitrary edges will be sketched in the following informal manner:



We distinguish whether $ctx(v)$ is positive or negative, and we start with the case that $ctx(v)$ is positive. Then we have



This derivation can be carried out in arbitrary positive contexts. Thus, the inverse derivation (from the right to the left, using the rules specialization, identity-erasure and ⊤-erasure) can be carried out in arbitrary negative contexts.[1] If otherwise $ctx(v)$ is negative, we have



This derivation can be carried out in arbitrary negative contexts. A similar argumentation to the first case yields that the inverse derivation can be carried out in arbitrary positive contexts.

From both cases we conclude that $\mathfrak{G}$ and $\mathfrak{G}'$ are provably equivalent. □

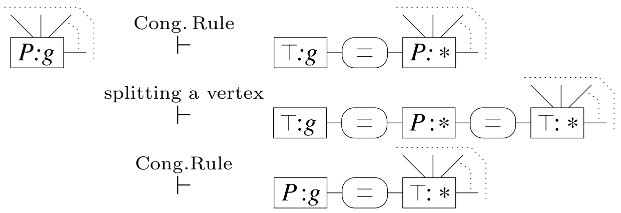
From Lem. 10.3 we can immediately conclude the following corollary, which is a combination of the Congruence Rule and the 'splitting a vertex'-rule. The crucial point is that, in contrast to the 'splitting a vertex'-rule, $v = \boxed{P : g}$ is substituted by $\boxed{\top : *}$ and not by $\boxed{\top : g}$, even if $g \neq *$.

Corollary 10.4.

Let $\mathfrak{G} := (V, E, \nu, \top, Cut, area, \kappa, \rho)$ be a concept graph. Let $g \in \mathcal{G} \cup \{*\}$ and let $v = \boxed{P : g}$ be a vertex in a context c. Then the following may be done: In c, a new vertex $v' = \boxed{P : g}$ and a new identity link between v and v' is inserted. Then $v = \boxed{P : g}$ is substituted by $\boxed{\top : *}$.

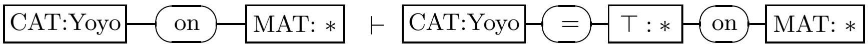
[1] This is an analogous argumentation to the proof of the Cut-and-Paste Theorem, i.e., Lem. 5.4, in the part Alpha.

Proof: Let $v = \boxed{P : g}$ be a vertex in an arbitrary context with $g \in \mathcal{G} \cup \{*\}$ and $P \in \mathcal{C}$. Then we can do the following:



As this derivation can be carried out in both directions, the proof is finished. □

Again we provide an example for the procedure described in the lemma.



Corollary 10.4 allows us in some proofs to assume w.l.o.g. that some vertices we consider have the form $\boxed{\top : *}$. An example for this can be already found in the proof of the next lemma.

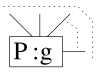
After proving two congruence results, we will now focus on the treatment of identity. Identity is a reflexive, transitive, and symmetric relation. In the calculus we use for FOL (see Fig. 8.1), these properties are directly captured through the axioms Id1, Id2 and Id3. Identity links in concept graphs are treated differently, and the axioms Id1, Id2 and Id3 have no corresponding rules in the calculus for concept graphs. Nevertheless, is has to be expected that identity links can be handled in a reflexive, transitive, and symmetric manner. This is formalized and proven in the next lemma.

Lemma 10.5 (Id.-Links are Reflexive, Transitive, Symmetric).
Identity links are reflexive, transitive, and symmetric, i.e. the following operations may be performed:

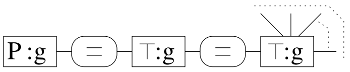
- **Reflexivity:** *Let $v \in V$ be an arbitrary vertex. Then an identity link e with $\kappa(e) = \doteq$, $ctx(e) = ctx(v)$ and $e\big|_1 = e\big|_2 = v$ may be inserted or erased.*
- **Symmetry:** *Let e be an edge with $\kappa(e) = \doteq$. Then e may be substituted by an edge e' which satisfies $e'\big|_1 = e\big|_2$, $e'\big|_2 = e\big|_1$ and $ctx(e') = ctx(e)$ (i.e., the orientation of the edge is reversed).*
- **Transitivity:** *Let e_1, e_2 be two edges with $\kappa(e_1) = \kappa(e_2) = \doteq$, $ctx(e_1) = ctx(e_2)$ and $e_1\big|_2 = e_2\big|_1$. Then an edge e with $\kappa(e) = \doteq$, $ctx(e) = ctx(e_1)$, $e\big|_1 = e_1\big|_1$ and $e\big|_2 = e_2\big|_2$ may be inserted or erased.*

Proof: We start with the proof for reflexivity.

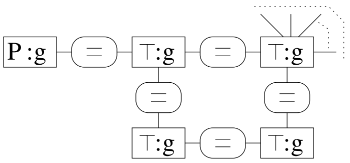
Let $\boxed{P{:}g}$, $g \in \mathcal{G} \cup \{*\}$ be a concept box in an arbitrary context (incident with arbitrary edges):



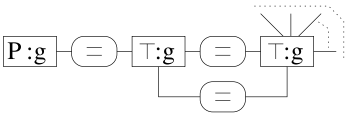
A twofold application of the rule 'splitting a vertex' yields:



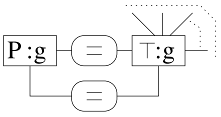
Iteration yields:



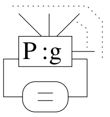
Now the two new vertices $\boxed{\top{:}g}$ are merged into their counterparts:



The left concept box $\boxed{\top{:}g}$ is merged back into $\boxed{P{:}g}$:



The remaining concept box $\boxed{\top{:}g}$ is merged back into $\boxed{P{:}g}$:

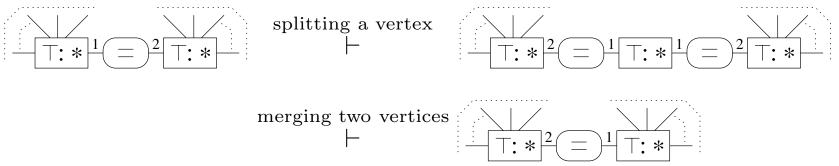


Note that the whole proof can be carried out in both directions, hence the proof of the reflexivity is done.

We proceed with the proof for symmetry. Corollary 10.4 allows us to assume that $e\big|_1 = e|_2 = \boxed{\top : *}$ (if for example $e|_1 = \boxed{P : g} \neq \boxed{\top : *}$, apply the congruence rule to transform $e|_1$ into $\boxed{\top : *}\!-\!(=)\!-\!\boxed{P : g}$, carry out the proof and transform $\boxed{\top : *}\!-\!(=)\!-\!\boxed{P : g}$ back into $\boxed{P : g}$).

Furthermore we can assume that we have $ctx(e|_1) = ctx(e|_2) = ctx(e)$ (if for example $ctx(e|_1) > ctx(e)$, split the vertex $e|_1$ such that the new vertex $\boxed{\top : *}$ is placed in $ctx(e)$, carry out the proof and apply the rule 'merging two vertices' to reverse the splitting at the beginning).

Now let $v_1 = \boxed{\top : *}$ and $v_2 = \boxed{\top : *}$ be two vertices and let e be an identity link with $e|_1 = v_1$, $e|_2 = v_2$ and $ctx(e|_1) = ctx(e|_2) = ctx(e)$. Then we have



which finishes the proof for symmetry.

The proof for transitivity relies on the same ideas, hence it is omitted here.□

We want to point out that Lem. 10.5 allows us to omit the labelling of identity links in the rest of this treatise.

10.3 Soundness of the Calculus

Like in the proof for the soundness of the calculus in the part Alpha of this treatise, the main concept for this proof is the concept of an isomorphism except a context (see Def. 2.13). The use of the generic marker forced the definition and use of partial and total valuations which assign objects to generic nodes when we evaluate a concept graph. Valuations are canonically transferred by an isomorphism except a context. This is captured by the following definition.

Definition 10.6 (Partial Isomorphism Applied to Valuations).

Let $\mathfrak{G} := (V, E, \nu, \top, Cut, area, \kappa, \rho)$, $\mathfrak{G}' := (V', E', \nu', \top', Cut', area', \kappa', \rho')$ be concept graphs with cuts and let f be an isomorphism between $\mathfrak{G}$ and $\mathfrak{G}'$ except for $c \in Cut$ and $c' \in Cut'$. Let ref be a partial valuation on $\mathfrak{G}$ such that $dom(ref) \cap \{v \in V^* \,|\, ctx(v) \leq c\} = \emptyset$. Then we define $f(ref)$ on $\{f(v) \,|\, v \in V \cap dom(ref)\}$ by $f(ref)(f(v)) := ref(v)$.

Now we can start with the proof of the soundness. The main theorem for the whole proof is the following:

Theorem 10.7 (Main Thm. for Soundness, Implication Version).

Let $\mathfrak{G} := (V, E, \nu, \top, Cut, area, \kappa, \rho)$, $\mathfrak{G}' := (V', E', \nu', \top', Cut', area', \kappa', \rho')$ be two graphs and let f be an isomorphism between $\mathfrak{G}$ and $\mathfrak{G}'$ except for the cuts $c \in Cut$ and $c' \in Cut'$. Let $D \subseteq Cut \cup \{\top\}$ be a set of contexts with $c \in D$ and which fulfills for all contexts d, e the following implication: $d \in D \wedge e \leq d \wedge e \not< c \Rightarrow e \in D$. Let ref_0 be a partial valuation which fulfills $dom(ref_0) \supseteq \{v \in V^* \,|\, ctx(v) \notin D \wedge \exists d \in D \,.\, ctx(v) > d\}$ and which fulfills $dom(ref_0) \cap \{v \in V^* \,|\, \exists d {\in} D \,.\, ctx(v) \leq d\} = \emptyset$.

Now let $Q(d)$ be the following property for contexts $d \in D$:

- *If d is positive and $ref \supseteq ref_0$ is a partial valuation for d such that $(\vec{\mathbb{K}}, \lambda) \models_{endo} \mathfrak{G}[d, ref]$ then $(\vec{\mathbb{K}}, \lambda) \models_{endo} \mathfrak{G}'[f(d), f(ref)]$*

– *If d is negative and $ref \supseteq ref_0$ is a partial valuation for d such that $(\vec{\mathbb{K}},\lambda) \not\models_{endo} \mathfrak{G}[d, ref]$ then $(\vec{\mathbb{K}},\lambda) \not\models_{endo} \mathfrak{G}'[f(d), f(ref)]$*

If Q holds for c, then Q holds for each $d \in D$.

Proof: In Fig. 10.1 we have sketched one situation in which this theorem will be applied. In this situation, we have a cut $c_0 > c$, and the set D contains all cuts below c_0 which are not below c. This situation will occur when in the graph $\mathfrak{G}$ a subgraph is iterated from the cut c_0 into the cut c. (Furthermore results from this situation the name ref_0 for the valuation which assigns objects to the concept boxes above D.)

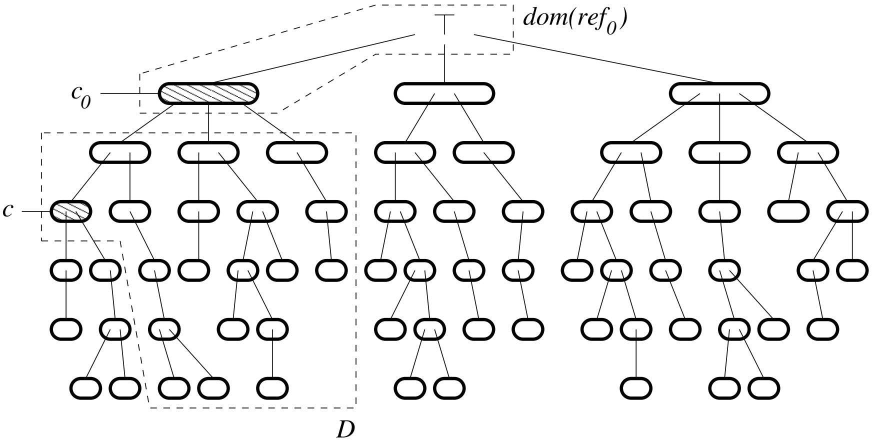


Fig. 10.1. The situation when a subgraph is iterated from c_0 into c

Proof: We set $D := \{d \in Cut \cup \{\top\} \mid d \not< c\}$. D is a tree such that for each $d \in D$ with $d \neq c$ and each $e \in Cut \cup \{\top\}$ with $e < d$ we have $e \in D$. For this reason we can carry out the proof by induction over D. So let $d \in D$, $d \neq c$ (we know that c satisfies Q) be a context such that $Q(e)$ holds for all cuts $e \in area(d) \cap Cut$. Furthermore let ref be a partial valuation for d.

Like in the proof of Thm. 6.1 in the part Alpha of this treatise, there are two cases to consider:

– **First Case:** d is positive and $(\vec{\mathbb{K}},\lambda) \models_{endo} \mathfrak{G}[d, ref]$.

As we have $(\vec{\mathbb{K}},\lambda) \models_{endo} \mathfrak{G}[d, ref]$, we can extend ref to a mapping $\widetilde{ref}$: $dom(ref) \cup (V \cap area(d)) \to G_0$ such that all vertex- and edge-conditions in d hold and such that $(\vec{\mathbb{K}},\lambda) \not\models_{endo} \mathfrak{G}[e, \widetilde{ref}]$ for all cuts $e \in area(d) \cap Cut$. Like in Def. 10.6, $f(ref)$ can be canonically extended to $\widetilde{f(ref)}$ such that we have $\widetilde{f(ref)} = f(\widetilde{ref})$. The isomorphism yields that all edge and vertex conditions hold for $\widetilde{f(ref)}$, and the induction hypothesis yields that $(\vec{\mathbb{K}}, \lambda) \not\models_{endo} \mathfrak{G}'[f(e), \widetilde{f(ref)}]$ for all cuts $e \in area(d) \cup Cut$. Hence we have $(\vec{\mathbb{K}}, \lambda) \models_{endo} \mathfrak{G}'[f(d), f(ref)]$.

- **Second Case:** d is negative and $(\vec{\mathbb{K}}, \lambda) \not\models_{endo} \mathfrak{G}[d, ref]$.

  Assume that we have $(\vec{\mathbb{K}}, \lambda) \models_{endo} \mathfrak{G}'[f(d), f(ref)]$, i.e., $f(ref)$ can be extended to $\widetilde{f(ref)}$ such that all vertex and edge conditions hold in $f(d)$ and such that $(\vec{\mathbb{K}}, \lambda) \not\models_{endo} \mathfrak{G}'[e', \widetilde{f(ref)}]$ for all cuts $e' \in area(f(d)) \cap Cut'$. Obviously there exists an extension $\widetilde{ref}$ of ref such that $f(\widetilde{ref}) = \widetilde{f(ref)}$. We conclude that all edge and vertex conditions hold for $\widetilde{ref}$ in d. Our assumption yields that we have $(\vec{\mathbb{K}}, \lambda) \not\models_{endo} \mathfrak{G}'[f(e), \widetilde{f(ref)}]$, i.e., $(\vec{\mathbb{K}}, \lambda) \not\models_{endo} \mathfrak{G}'[f(e), f(\widetilde{ref})]$ for all cuts $e \in area(d)$. By induction hypothesis we have $(\vec{\mathbb{K}}, \lambda) \not\models_{endo} \mathfrak{G}[e, \widetilde{ref}]$ for all cuts $e \in area(d)$. So $\widetilde{ref}$ is an extension of ref which fulfills all properties of Def. 9.3, hence we have $(\vec{\mathbb{K}}, \lambda) \models_{endo} \mathfrak{G}[d, ref]$, a contradiction.

From both cases we conclude the theorem. □

This is the main theorem to prove the soundness of those rules which weaken the informational content[2] of a graph (e.g. erasure and insertion). Other rules (e.g. iteration and deiteration) do not change the informational content and may therefore be performed in both directions. For proving the soundness of those rules, a second version of this theorem is needed which does not distinguish between positive and negative contexts and which uses an equivalence instead of two implications for the property Q.

Theorem 10.8 (Main Thm. for Soundness, Equivalence Version). *Let $\mathfrak{G} := (V, E, \nu, \top, Cut, area, \kappa, \rho)$, $\mathfrak{G}' := (V', E', \nu', \top', Cut', area', \kappa', \rho')$ be two graphs and let f be an isomorphism between $\mathfrak{G}$ and $\mathfrak{G}'$ except for the cuts $c \in Cut$ and $c' \in Cut'$. Let $D \subseteq Cut \cup \{\top\}$ be a set of contexts with $c \in D$ and which fulfills for all contexts d, e the following implication: $d \in D \wedge e \leq d \wedge e \not< c \Rightarrow e \in D$. Let ref_0 be a partial valuation which fulfills $dom(ref_0) \supseteq \{v \in V^* \mid ctx(v) \notin D \wedge \exists d \in D \,.\, ctx(v) > d\}$ and which fulfills $dom(ref_0) \cap \{v \in V^* \mid \exists d {\in} D \,.\, ctx(v) \leq d\} = \emptyset$.*

[2] A precise definition of informational content, which is called *conceptual content*, can be found in [79], [80]. Here it is sufficient to use a naive understanding of this term.

Now let $Q(d)$ be the following property for contexts $d \in D$:

Every partial valuation ref for d with $ref \supseteq ref_0$ satisfies

$$(\vec{\mathbb{K}}, \lambda) \models_{endo} \mathfrak{G}[d, ref] \quad \Longleftrightarrow \quad (\vec{\mathbb{K}}, \lambda) \models_{endo} \mathfrak{G}'[f(d), f(ref)]$$

If Q holds for c, then Q holds for each $d \in D$.

Proof: Done analogously to the proof of Thm. 10.7. □.

The next two corollaries follow immediately from the last two theorems by setting $D := \{d \in Cut \cup \{\top\} \mid d \not< c\}$ (hence $dom(ref_0) = \emptyset$). Please note that Cor. 10.9 is the Beta-version of Thm. 6.1. In Lem. 10.13 we will need Thm. 10.8 as well as Cor. 10.10. To avoid confusion in the proof of Lem. 10.13, we denote the Property in Corollaries 10.9 and 10.10 by P and not again by Q.

Corollary 10.9 (Main Cor. for Soundness, Implication Version). *Let $\mathfrak{G} := (V, E, \nu, \top, Cut, area, \kappa, \rho)$, $\mathfrak{G}' := (V', E', \nu', \top', Cut', area', \kappa', \rho')$ be two graphs and let f be an isomorphism between $\mathfrak{G}$ and $\mathfrak{G}'$ except for $c \in Cut$ and $c' \in Cut'$. Set $Cut_c := \{d \in Cut \cup \{\top\} \mid d \not< c\}$. Let $(\vec{\mathbb{K}}, \lambda)$ be a contextual model and let $P(d)$ be the following property for contexts $d \in Cut_c$:*

- *If d is a positive context and ref is a partial valuation for d which fulfills $(\vec{\mathbb{K}}, \lambda) \models_{endo} \mathfrak{G}[d, ref]$, then $(\vec{\mathbb{K}}, \lambda) \models_{endo} \mathfrak{G}'[f(d), f(ref)]$, and*
- *if d is a negative context and ref is a partial valuation for d which fulfills $(\vec{\mathbb{K}}, \lambda) \not\models_{endo} \mathfrak{G}[d, ref]$, then $(\vec{\mathbb{K}}, \lambda) \not\models_{endo} \mathfrak{G}'[f(d), f(ref)]$.*

If P holds for c, then P holds for each $d \in Cut_c$.

Corollary 10.10 (Main Cor. for Soundness, Equivalence Version). *Let $\mathfrak{G} := (V, E, \nu, \top, Cut, area, \kappa, \rho)$, $\mathfrak{G}' := (V', E', \nu', \top', Cut', area', \kappa', \rho')$ be two graphs and let f be an isomorphism between $\mathfrak{G}$ and $\mathfrak{G}'$ except for $c \in Cut$ and $c' \in Cut'$. Set $Cut_c := \{d \in Cut \cup \{\top\} \mid d \not< c\}$. Let $(\vec{\mathbb{K}}, \lambda)$ be a contextual model and let $P(d)$ be the following property for contexts $d \in Cut_c$:*

Every partial valuation ref for d satisfies

$$(\vec{\mathbb{K}}, \lambda) \models_{endo} \mathfrak{G}[d, ref] \quad \Longleftrightarrow \quad (\vec{\mathbb{K}}, \lambda) \models_{endo} \mathfrak{G}'[f(d), f(ref)]$$

If P holds for c, then P holds for each $d \in Cut_c$.

Now we can start to prove the soundness of all rules. We begin with the rules erasure and insertion. This lemma corresponds to Lem. 6.3 in the part Alpha of this treatise.

Lemma 10.11 (Erasure and Insertion are Sound).

If $\mathfrak{G}$ and $\mathfrak{G}'$ are two concept graphs with cuts, $(\vec{\mathbb{K}}, \lambda)$ is a contextual structure with $(\vec{\mathbb{K}}, \lambda) \models \mathfrak{G}$ and $\mathfrak{G}'$ is derived from $\mathfrak{G}$ by applying the rule 'erasure' or 'insertion', then $(\vec{\mathbb{K}}, \lambda) \models \mathfrak{G}'$.

Proof: We only show the soundness of the erasure-rule.

Let $\mathfrak{G}_0 := (V_0, E_0, \nu_0, \top_0, Cut_0, area_0)$ be the subgraph which is erased. $\mathfrak{G}_0$ is erased from the area of the positive context $c := \top_0$. Obviously, $\mathfrak{G}$ and $\mathfrak{G}'$ are isomorphic except for the context c by the (trivial) identity mapping. Let ref be a partial valuation for c such that $(\vec{\mathbb{K}}, \lambda) \models_{endo} \mathfrak{G}[c, ref]$. It is easy to see that we have $(\vec{\mathbb{K}}, \lambda) \models_{endo} \mathfrak{G}'[f(c), ref]$. So the property P of Cor. 10.9 holds for c, hence Cor. 10.9 can be applied. This yields the proposition.

The proof for the insertion-rule is done analogously. □

Obviously, the idea of this proof is the same as in the proof for Lem. 6.3 in the part Alpha of this treatise. This is the same for all rules except for iteration and deiteration. So we come immediately to the following lemma:

Lemma 10.12 (Further Rules are Sound).

Let $\mathfrak{G}$ and $\mathfrak{G}'$ be two concept graphs with cuts, let $(\vec{\mathbb{K}}, \lambda)$ be a contextual structure with $(\vec{\mathbb{K}}, \lambda) \models \mathfrak{G}$ and let $\mathfrak{G}'$ be derived from $\mathfrak{G}$ by applying one of the following rules:

- *double cuts*
- *generalization or specialization*
- *isomorphism*
- *exchanging references*
- *merging two vertices or splitting a vertex*
- *$\top$-erasure or $\top$-insertion*
- *identity-erasure or identity-insertion*

Then $(\vec{\mathbb{K}}, \lambda) \models \mathfrak{G}'$.

Proof: Using Corollaries 10.9 and 10.10, the proofs are done analogously to the proofs of the appropriate lemmata in the part Alpha of this treatise. □

Unfortunately, the proof for the soundness of the rules iteration and its counterpart deiteration is much more complex than in the Alpha-part of this treatise. The main reason is the following: Let $\mathfrak{G}_0$ be a subgraph of a graph $\mathfrak{G}$ which is iterated into a context c. Particulary (if $\mathfrak{G}_0$ contains generic nodes $\boxed{P : *}$ on its sheet of asseretion), new generic nodes $\boxed{P : *}$ are added to c. When we now evaluate the graph in a model with the endoporeutic method of Peirce, we have to assign objects to these nodes. The assignment of the 'right' objects will depend on which objects we have already assigned to generic nodes which are placed in the same or higher context as the new nodes. But the new nodes are copies of already existing nodes (their origins of $\mathfrak{G}_0$). Each time when we reach the new nodes while performing the endoporeutic method, we have already assigned objects to their origins. It turns out that we should assign the same objects to the old and new nodes to gain that the old and the new graph become equivalent. This idea is worked out in the proof for the following lemma.

Lemma 10.13 (Iteration and Deiteration are Sound).
If $\mathfrak{G}$ and $\mathfrak{G}'$ are two concept graphs with cuts, $(\vec{\mathbb{K}}, \lambda)$ is a contextual structure with $(\vec{\mathbb{K}}, \lambda) \models \mathfrak{G}$ and $\mathfrak{G}'$ is derived from $\mathfrak{G}$ by applying the rule 'iteration' or 'deiteration', then $(\vec{\mathbb{K}}, \lambda) \models \mathfrak{G}'$.

Proof: Let $\mathfrak{G} := (V, E, \nu, \top, Cut, area, \kappa, \rho)$ be a concept graph with the (not necessarily closed) subgraph $\mathfrak{G}_0 := (V_0, E_0, \nu_0, \top_0, Cut_0, area_0, \kappa_0, \rho_0)$. Let $\mathfrak{G}' := (V', E', \nu', \top', Cut', area', \kappa', \rho')$ be a graph which is derived from $\mathfrak{G}$ by iterating $\mathfrak{G}_0$ into the context c. Consider the canonical mapping f which is defined by $f(k) = (k, 1)$ for $k \in E \cup V \cup Cut$. Then $\mathfrak{G}$ and $\mathfrak{G}'$ are isomorphic up to $c_0 := \top_0$ and $(c_0, 1)$ by f. They are even isomorphic up to c and $(c, 1)$. We want to apply Cor. 10.10 to $\mathfrak{G}$, c_0 and $\mathfrak{G}'$, $(c_0, 1)$, so let ref be a partial valuation for c_0. Then we have a corresponding partial valuation $f(ref)$ for $f(c_0) = (c_0, 1)$. We have to show that the property P of Cor. 10.10 holds for c_0. In order to do this, we distinguish two cases:

- **First case:** $c = c_0$

  We want to prove

  $$(\vec{\mathbb{K}}, \lambda) \models_{endo} \mathfrak{G}[c_0, ref] \quad \Longleftrightarrow \quad (\vec{\mathbb{K}}, \lambda) \models_{endo} \mathfrak{G}'[(c_0, 1), f(ref)] \qquad (10.1)$$

  i.e., we have to show the following:

  - For each extension $\widetilde{ref}$ of ref on $area(c_0)$ with $(\vec{\mathbb{K}}, \lambda) \models_{endo} \mathfrak{G}[c_0, \widetilde{ref}]$ exists an extension $\widetilde{f(ref)}$ of $f(ref)$ on $area'((c_0, 1))$ with $(\vec{\mathbb{K}}, \lambda) \models_{endo} \mathfrak{G}'[(c_0, 1), \widetilde{f(ref)}]$, and
  - Vice versa: For each extension $\widetilde{f(ref)}$ of $f(ref)$ on $area'((c_0, 1))$ with $(\vec{\mathbb{K}}, \lambda) \models_{endo} \mathfrak{G}'[(c_0, 1), \widetilde{f(ref)}]$ exists an extension $\widetilde{ref}$ of ref on $area(c_0)$ with $(\vec{\mathbb{K}}, \lambda) \models_{endo} \mathfrak{G}[c_0, \widetilde{ref}]$.

  We start with '$\Longrightarrow$' of (10.1), i.e., we assume that ref is a partial valuation for c_0 which fulfills $(\vec{\mathbb{K}}, \lambda) \models_{endo} \mathfrak{G}[c_0, ref]$. This means that we can extend ref to $\widetilde{ref}$ on $area(c_0)$ (i.e., $dom(\widetilde{ref}) = dom(ref) \cup \{v \in V^* \mid ctx(v) = c_0\}$) such that $(\vec{\mathbb{K}}, \lambda) \models_{endo} \mathfrak{G}[c_0, \widetilde{ref}]$ holds.

We can extend $f(ref)$ on $area'((c_0,1))$ in the same way as ref: For each $(v,i) \in area((c_0,1))$ (hence $v \in area(c_0)$) let $\widetilde{f(ref)}((v,i)) := \widetilde{ref}(v)$. It is easy to see that we have $(\vec{\mathbb{K}},\lambda) \models_{endo} \mathfrak{G}'[(c_0,1),\widetilde{f(ref)}]$, thus we have $(\vec{\mathbb{K}},\lambda) \models_{endo} \mathfrak{G}'[(c_0,1),f(ref)]$. In particular '$\Longrightarrow$' of (10.1) holds.

Assume on the other hand $(\vec{\mathbb{K}},\lambda) \models_{endo} \mathfrak{G}'[(c_0,1),f(ref)]$. Then we can extend $f(ref)$ to $\widetilde{f(ref)}$ such that $(\vec{\mathbb{K}},\lambda) \models_{endo} \mathfrak{G}'[(c_0,1),\widetilde{f(ref)}]$ holds. Now extend ref to $\widetilde{ref}$ by $\widetilde{ref}(v) := \widetilde{f(ref)}((v,1))$. It is easy to see that we have $(\vec{\mathbb{K}},\lambda) \models_{endo} \mathfrak{G}[c_0,\widetilde{ref}]$, hence we have $(\vec{\mathbb{K}},\lambda) \models_{endo} \mathfrak{G}[c_0,ref]$. Hence '$\Longleftarrow$' of (10.1) holds.

As we have shown (10.1), we conclude that P holds for c_0.

- **Second case:** $c \lneq c_0$

  Let ref_0 be an extension of ref on $area(c_0)$. We want to show that we have

  $$(\vec{\mathbb{K}},\lambda) \models_{endo} \mathfrak{G}[c_0,ref_0] \quad \Longleftrightarrow \quad (\vec{\mathbb{K}},\lambda) \models_{endo} \mathfrak{G}'[(c_0,1),f(ref_0)]. \tag{10.2}$$

  The context c_0 contains the vertices, edges and cuts which are written in the subgraph $\mathfrak{G}_0$ on its sheet of assertion (i.e., c_0), but there may be more vertices, edges and cuts in c_0. So in order to check whether we have $(\vec{\mathbb{K}},\lambda) \models_{endo} \mathfrak{G}[c_0,ref]$, we have particularly to check whether all vertex-conditions for vertices $v \in V_0 \cap area_0(c_0)$, all edge-conditions for edges $e \in E_0 \cap area_0(c_0)$ and all cut-conditions for all cuts $d \in Cut_0 \cap area(\top_0)$ hold. In this proof we will denote by $(\vec{\mathbb{K}},\lambda) \models_{endo} \mathfrak{G}_0[c_0,ref_0|_{V_0}]$ that all these conditions hold.

  If we have $(\vec{\mathbb{K}},\lambda) \not\models_{endo} \mathfrak{G}_0[c_0,ref_0|_{V_0}]$, it is easy to see that we have $(\vec{\mathbb{K}},\lambda) \not\models_{endo} \mathfrak{G}[c_0,ref_0]$ and $(\vec{\mathbb{K}},\lambda) \not\models_{endo} \mathfrak{G}[(c_0,1),f(ref_0)]$. Hence (10.2) is fulfilled. So we assume that $(\vec{\mathbb{K}},\lambda) \models_{endo} \mathfrak{G}_0[c_0,ref_0|_{V_0}]$ holds. We want to apply Thm. 10.8 to the set $D := \{d \in Cut \cup \{\top\} \mid d < c_0 \wedge d \not< c\}$, so we have to show that property Q of Cor. 10.8 holds for c. This is done similar to the first case. Let ref_c be a partial valuation for c which extends ref_0. Let $\widetilde{ref_c}$ be an extension of ref_c to $area(c)$. Note that in the graph $\mathfrak{G}'$, the appropriate cut $(c,1)$ contains two kinds of vertices: 'Old' vertices $(v,1)$ whose counterparts v in $\mathfrak{G}$ are scribed on the area of c, and 'new' vertices $(v,2)$ whose counterparts v in $\mathfrak{G}$ are scribed on the area of c_0. For both kinds of vertices we adopt the valuation of $\widetilde{ref_c}$, i.e., for $(v,i) \in area((c,1))$, we set $\widetilde{f(ref_c)}((v,i)) := \widetilde{ref_c}(v)$. Then $\widetilde{f(ref_c)}$ is an extension of $f(ref_c)$ to $area((c,1))$.

  As we have $(\vec{\mathbb{K}},\lambda) \models_{endo} \mathfrak{G}_0[c_0,ref_0|_{V_0}]$, it is easy to see that the copy of the subgraph $\mathfrak{G}_0$ which is iterated into c and the supplementary identity links which are inserted into c evaluate under $\widetilde{f(ref_c)}$ to true. So we have only to check the edges, vertices and cuts which appear in c as well as in $(c,1)$. This yields

$$(\vec{\mathbb{K}},\lambda) \models_{endo} \mathfrak{G}[c,\widetilde{ref_c}] \quad\Longleftrightarrow\quad (\vec{\mathbb{K}},\lambda) \models_{endo} \mathfrak{G}'[(c,1),\widetilde{f(ref_c)}] \quad .$$

Similar to the argumentation after (10.1) in the first case, this equivalence yields that property Q of Cor. 10.8 holds for c. Hence Cor. 10.8 can be applied. This yields that for every cut $d \in area(c_0)$ we have

$$(\vec{\mathbb{K}},\lambda) \models_{endo} \mathfrak{G}[d,ref_0] \quad\Longleftrightarrow\quad (\vec{\mathbb{K}},\lambda) \models_{endo} \mathfrak{G}'[(d,1),f(ref_0)] \quad .$$

Furthermore we have the same vertex- and edge-conditions in c and $(c,1)$. Hence this equivalence yields that (10.2) holds. Again we need an argumentation like in the first case to conclude from (10.2) that property P of Cor. 10.10 holds for c_0.

Now we can apply Cor. 10.10 to $\mathfrak{G}$, c_0 and $\mathfrak{G}_0$, $(c_0,1)$. This yields

$$(\vec{\mathbb{K}},\lambda) \models_{endo} \mathfrak{G} \quad\Longleftrightarrow\quad (\vec{\mathbb{K}},\lambda) \models_{endo} \mathfrak{G}' \quad ,$$

which shows the soundness of both the iteration- and deiteration-rule. □.

Like in the part Alpha, we conclude from the preceding lemmata the soundness of the calculus:

Theorem 10.14 (Soundness of the Beta-Calculus).

A set of concept graphs $\mathfrak{H}$ and a concept graph $\mathfrak{G}$ with cuts over $\mathcal{A}$ satisfy

$$\mathfrak{H} \vdash \mathfrak{G} \quad\Longrightarrow\quad \mathfrak{H} \models \mathfrak{G}$$

Proof: This theorem follows immediately from Lem. 10.11, Lem. 10.12 and Lem. 10.13. □

11 Syntactical Equivalence to FOL

In Chap. 9 we provided two translations $\Phi : \mathrm{CG} \to \mathrm{FOL}$ and $\Psi : \mathrm{FOL} \to \mathrm{CG}$ between concept graphs and first order logic. Both systems are equipped with a semantical entailment relation $\models$ and a derivability relation $\vdash$. In Chap. 9 we have already shown that Φ respects the entailment relation $\models$ (see Thm. 9.10). In this chapter we will focus on the derivability relations $\vdash$.

In the first section we will show the following: If we have a formula f, translate it to a concept graph $\Psi(f)$, and translate this concept graph back to a formula $f' := \Phi(\Psi(f))$, then f' and f are provably equivalent (but usually, f and f' are not identical).

It has to be expected that the same holds in the opposite direction for concept graphs, i.e., that we have $\mathfrak{G} \vdash \Psi(\Phi(\mathfrak{G}))$ and $\Psi(\Phi(\mathfrak{G})) \vdash \mathfrak{G}$. This will be proven in the next section.

In the last section we will show that Ψ respects the derivability relation $\vdash$, i.e., we have $f_1 \vdash f_2 \Longrightarrow \Psi(f_1) \vdash \Psi(f_2)$. The opposite direction for concept graphs, i.e. $\mathfrak{G}_1 \vdash \mathfrak{G}_2 \Longrightarrow \Phi(\mathfrak{G}_1) \vdash \Phi(\mathfrak{G}_2)$, holds too. But it will turn out in Lem. 12.1 that this is an immediate consequence from the fact that both calculi (on FOL and on CG) are sound and on the fact that Φ respects the semantical entailment relation $\models$. For this reason we do not have to show explicitly in this chapter that Φ respects $\vdash$.

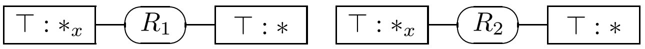
11.1 Equivalence of f and $\Phi(\Psi(f))$ for Formulas f

Please remember that for each concept graph $\mathfrak{G}$, its translation $\Phi(\mathfrak{G})$ to FOL is not uniquely defined, but only given up to the relation $\cong$ (see Definition of Φ and Def. 9.5). Hence, if f is a formula, $\Phi(\Psi(f))$ can only be given up to $\cong$ (in particular it is not possible to prove that $\Phi \circ \Psi$ is the identity-mapping on FOL). There may be even more differences between f and $\Phi(\Psi(f))$ which are not captured by the relation $\cong$, i.e., we will usually have $f \ncong \Phi(\Psi(f))$ for formulas f. Nevertheless, we expect that the formulas f and $\Phi(\Psi(f))$ will not be substantially different. In particular we expect f and $\Phi(\Psi(f))$ to be provably equivalent. This will be worked out in this section.

F. Dau: The Logic System of Concept Graphs with Negation, LNAI 2892, pp. 125-156, 2003.

To give an impression how f and $\Phi(\Psi(f))$ may differ, we provide a small number of examples. As Φ is also defined for formulas with free variables, in these examples we have taken into account free variables.

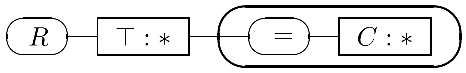
Example 11.1. Let $f_1 := (\exists y.xR_1y) \wedge (\exists y.\exists y.xR_2y)$. We have

$\Psi(f_1) =$  ,

hence $\Phi(\Psi(f_1))$ is the formula

$$\exists y.\exists z.(\top(x) \wedge \top(x) \wedge \top(y) \wedge \top(z) \wedge xR_1y \wedge xR_2z) \quad .$$

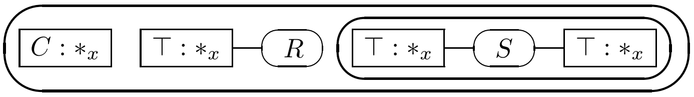
Example 11.2. Let $f_2 := \exists x.(R(x) \wedge \neg P(x))$ with $R \in \mathcal{R}_1$ and $C \in \mathcal{C}$. Then we have

$\Psi(f_2) =$  ,

hence $\Phi(\Psi(f_2))$ is the formula

$$\exists x.(\top(x) \wedge R(x) \wedge \neg\exists y.(x = y \wedge C(y)) \quad .$$

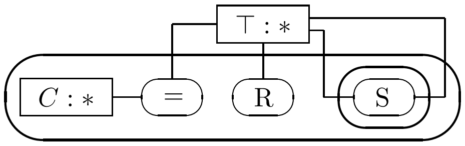
Example 11.3. Let $f_3 := \neg(C(x) \wedge R(x) \wedge \neg S(x,x))$. Then we have

$\Psi(f_3) =$  ,

hence $\Phi(\Psi(f_3))$ is the formula

$$\neg(C(x) \wedge \top(x) \wedge R(x) \wedge \neg(\top(x) \wedge S(x,x) \wedge \top(x))) \quad .$$

Example 11.4. Let $f_4 :== \exists x.\neg(C(x) \wedge R(x) \wedge \neg S(x,x))$. Note that f_4 is the existential closure of the formula f_3 in Example 3. Now we have

$\Psi(f_4) =$  ,

hence $\Phi(\Psi(f_4))$ is the formula

$$\exists x.(\top(x) \wedge \neg\exists y.(C(y) \wedge x = y \wedge R(x) \wedge \neg S(x,x)))) \quad .$$

The formula $C(x) \wedge R(x) \wedge \neg S(x,x)$ is a subformula of f_3 as well as of f_4. We want to point out that in Example 3, this subformula is converted to the subformula $\top(x) \wedge S(x,x) \wedge \top(x)$ of $\Phi(\Psi(f_3))$, but in Example 4, this subformula in converted to the subformula $\exists y.(C(y) \wedge x = y \wedge R(x) \wedge \neg S(x,x))$ of $\Phi(\Psi(f_4))$.

When transforming a formula f to the formula $\Phi(\Psi(f))$, the following modifications may happen (the first three modifications are already mentioned after the definition of Φ and captured by the equivalence relation $\cong$):

- the order of existential quantifiers may change,
- the order of subformulas of a conjunction may change,
- bound variables may be renamed,
- atomic subformulas $\top(\alpha)$ may be added for variables α,
- superfluous quantifiers may be erased (see Example 1),
- quantifiers may move (see Example 1), and
- a variable may be substituted by a number of other variables, which are equated in the formula (see Example 2).

These modifications are captured by the following relation $\simeq$, which is obviously a generalization of the relation $\cong$ (see Def. 9.5).

Definition 11.5 (Second Equivalence Relation for Formulas).

Let $\simeq$ be the smallest equivalence relation on FOL such that the following conditions hold:

1. *If f_1, f_2, f_3 are formulas, then we have $f_1 \wedge f_2 \simeq f_2 \wedge f_1$ and $(f_1 \wedge f_2) \wedge f_3 \simeq f_1 \wedge (f_2 \wedge f_3)$.*
2. *If f_1, f_2 are formulas with $f_1 \simeq f_2$ and if α, β are variables, then $\exists\alpha.\exists\beta.f_1 \simeq \exists\beta.\exists\alpha.f_2$.*
3. *If f and f' are equal up to renaming bound variables, then $f \simeq f'$.*
4. *If f is a formula and if t is a term, then $(f \wedge \top(t)) \simeq f$.*
5. *Let f be a formula, let $\alpha, \alpha_1, \ldots, \alpha_n$ be variables with $\alpha \in Free(f)$ and $\alpha_1, \ldots, \alpha_n \notin Var(f)$, and let f' be a formula that we gain from f if we replace each free occurence of α by one of the variables $\alpha, \alpha_1, \ldots, \alpha_n$. Then $\exists\alpha_1. \ldots .\exists\alpha_n.\alpha = \alpha_1 \wedge \ldots \wedge \alpha = \alpha_n \wedge f' \simeq f$. In particular we have the following equivalence:*[1] *$f \simeq \exists\beta.(\alpha = \beta \wedge f[\beta/\alpha])$ for $\beta \notin Var(f)$.*
6. *If f is a formula and if $\beta \notin Free(f)$ is a variable, then $\exists\beta.f \simeq f$.*
7. *If f, g are formulas and if $\alpha \notin Free(g)$ is a variable, then $\exists\alpha.(g \wedge f) \simeq (g \wedge \exists\alpha.f)$.*

[1] Compare this rule with Cor. 3 in Chap. 3.5. of [57].

8. If f_1, f_2, g_1, g_2 are formulas with $f_1 \simeq f_2$ and $g_1 \simeq g_2$, then $\neg f_1 \simeq \neg f_2$, $f_1 \wedge g_1 \simeq\ f_2 \wedge g_2$ and $\exists\alpha.f_1 \simeq \exists\alpha.f_2$.

The following lemma is well known and the proof is therefore omitted here.

Lemma 11.6.

Let f, g be two formulas with $f \simeq g$. Then we have $\vdash f \leftrightarrow g$. In particular we have $f \vdash g$ and $g \vdash f$.

Without proof.

The relation $\simeq$ captures exactly the modifications which may happen when a formula f is transformed to $\Phi(\Psi(f))$. This is subject of the following lemma and theorem. In the lemma, we prove it for variable-purified formulas (see Def. 8.4), but as each formula can be transformed into a variable-purified formula, the lemma can immediately be generalized to arbitrary formulas. This will be done in the preceeding theorem.

Lemma 11.7 ($f \simeq \Phi(\Psi(f))$ for Variable-Purified Formulas).
Each variable-purified formula $f \in FOL$ satisfies $f \simeq \Phi(\Psi(f))$.

Proof: Remember that we defined $\Phi(\mathfrak{G}, c)$ inductively for each concept graph $\mathfrak{G} := (V, E, \nu, \top, Cut, area, \kappa, \rho)$ and each context $c \in Cut \cup \{\top\}$. We show the following property for each formula $f \in$ FOL which is variable-purified:

$$g \simeq \Phi(\Psi(f), \Psi_{Neg}(g)) \text{ for each } g \in Neg_f \cup \{\top_f\} \tag{11.1}$$

Please note that g may contain free variables. The proof for (11.1) is now done by induction over the structure of formulas.

- If $f = C(t)$ for a term t and a concept name $C \in \mathcal{C}$, we have $\Phi(\Psi(f)) = f$ and $Neg_f \cup \{\top_f\} = \{f\}$, hence (11.1) holds.
- Let t_1, t_2 be two terms and $f = t_1{=}t_2$. Then we have

$$\Phi(\Psi(t_1{=}t_2)) \ = \ \top(t_1) \wedge \top(t_2) \wedge t_1{=}t_2 \simeq f$$

  and $Neg_f \cup \{\top_f\} = \{f\}$, hence (11.1) holds.
- Let $t_1, \ldots, t_n$ be n terms, let $R \in \mathcal{R}_n$ be a relation name and let $f = R(t_1, \ldots, t_n)$. Then we have

$$\Phi(\Psi(r(t_1, \ldots, t_n))) \ = \ \top(t_1) \wedge \ldots \wedge \top(t_n) \wedge r(t_1, \ldots, t_n) \simeq f$$

  and $Neg_f \cup \{\top_f\} = \{f\}$, hence (11.1) holds.

- Let f_1, f_2 be two formulas and $f = f_1 \wedge f_2$.

  Let $\Psi(f_i) = \mathfrak{G}_i = (V_i, E_i, \nu_i, \top_i, Cut_i, area_i, \kappa_i, \rho_i)$, $i = 1, 2$. In the definition of $\Phi(\mathfrak{G}_1\ \mathfrak{G}_2)$, we assign variables to the generic boxes of $\mathfrak{G}_1\ \mathfrak{G}_2$. We do the same in the definition of $\Phi(\mathfrak{G}_1)$ and $\Phi(\mathfrak{G}_2)$. It is sufficient to consider the case that we have chosen for each generic concept box $v \in \mathfrak{G}_i$, $i = 1, 2$, the same variable in the definition of $\Phi(\mathfrak{G}_i)$ and $\Phi(\mathfrak{G}_1\ \mathfrak{G}_2)$.

  Now let $c^i_1, \ldots, c^i_{k_i}$ be the cuts in $\mathfrak{G}_i$ with $ctx(c^i_n) = \top_i$. Furthermore, let $v^i_1, \ldots, v^i_{l_i}$ be the nodes in V_i with $ctx(v^i_n) = \top_i$. Then for each v^i_n we have a variable $\alpha^i_n \notin Free(\Psi(f_i))$, for each cut c^i_n w have a formula h^i_n, and we have two formulas g^1 and g^2, such that

  $$\Phi(\Psi(f_i)) = \exists\alpha^i_1 \ldots \exists\alpha^i_{k_i}(g^i \wedge \neg h^i_1 \wedge \ldots \wedge \neg h^i_{l_i}) \quad .$$

  Note that $\Psi(f_1 \wedge f_2)$ is the juxtaposition of $\Psi(f_1)$ and $\Psi(f_2)$. The cuts $c^i_n \in Cut_i$ with $area(c^i_n) = \top_i$ correspond one-to-one to the cuts (c, i) in $\Psi(f_1 \wedge f_2)$ with $area((c, i)) = \top$. In particular (11.1) holds for each formula $g \in Neg_f \cup \{\top_f\}$ with $g \neq f$. So it remains to show that f satisfies $f \simeq \Phi(\Psi(f), \Psi_{Neg}(f))$. We have

  $$\begin{aligned}\Phi(\Psi(f_1 \wedge f_2)) &= \exists\alpha^1_1 \ldots \exists\alpha^1_{k_1} \exists\alpha^2_1 \ldots \exists\alpha^2_{k_2} \\ &\quad (g^1 \wedge g^2 \wedge \neg h^1_1 \wedge \ldots \wedge \neg h^1_{l_1} \wedge \neg h^2_1 \wedge \ldots \wedge \neg h^2_{l_2}) \quad .\end{aligned}$$

  The definition of Φ yields that $\alpha^1_1, \ldots, \alpha^1_{k_1} \notin Free(g^2 \wedge \neg h^2_1 \wedge \ldots \wedge \neg h^2_{l_2})$ and analogously $\alpha^2_1, \ldots, \alpha^2_{k_2} \notin Free(g^1 \wedge \neg h^1_1 \wedge \ldots \wedge \neg h^1_{l_1})$. Thus the definition of $\simeq$ yields

  $$\begin{aligned}\Phi(\Psi(f_1 \wedge f_2)) &= \exists\alpha^1_1 \ldots \exists\alpha^1_{k_1} . \exists\alpha^2_1 \ldots \exists\alpha^2_{k_2} . \\ &\qquad (g^1 \wedge g^2 \wedge \neg h^1_1 \wedge \ldots \wedge \neg h^1_{l_1} \wedge \neg h^2_1 \wedge \ldots \wedge \neg h^2_{l_2}) \\ &\simeq \exists\alpha^1_1 \ldots \exists\alpha^1_{k_1}(g^1 \wedge \neg h^1_1 \wedge \ldots \wedge \neg h^1_{l_1}) \wedge \\ &\quad\ \exists\alpha^2_1 \ldots \exists\alpha^2_{k_2}(g^2 \wedge \neg h^2_1 \wedge \ldots \wedge \neg h^2_{l_2}) \\ &= \Phi(\Psi(f_1)) \wedge \Phi(\Psi(f_2)) \\ &\overset{\text{I.H.}}{\simeq} f_1 \wedge f_2\end{aligned}$$

- $f = \neg f'$ for a formula f'.

  This case can be done analogously to the last case.

- $f = \exists\alpha.f'$ for a formula f' and a variable α. If $\alpha \notin Free(f)$, we have $\Psi(\exists\alpha.f) = \Psi(f)$, hence

  $$\Phi(\Psi(\exists\alpha.f)) = \Phi(\Psi(f)) \overset{\text{IV}}{\simeq} f \overset{\text{Def} \simeq}{\simeq} \exists\alpha.f$$

  So let $\alpha \in Free(f)$. We set $\Psi(f) := \mathfrak{G} := (V, E, \nu, \top, Cut, area, \kappa, \rho)$ and $\Psi(\exists\alpha.f) := \mathfrak{G}^{f'} := (V', E', \nu', \top', Cut', area', \kappa', \rho')$. The definition of Ψ yields that we have a new vertex $v \notin V'$ such that

$$V = (V' \cup \{v\}) \backslash \{w \in V' \mid \rho'(w) = *_\alpha \wedge (\exists e \in E'. \exists i \in \mathbb{N}. \nu'(e)\big|_i = w)\}$$

We first prove that (11.1) holds for each $g \in Neg_f \cup \{\top_f\}$, $g \neq f$ by induction on the order on $Neg_f \cup \{\top_f\}$. So let $g \in Neg_f$ be a subformula of f' such that (11.1) holds for all $h \in Neg_f \cup \{\top_f\}$ with $h < g$. The induction hypothesis of the outer induction for f' yields $\bar{g} := \Phi(\Psi(f'), \Psi_{Neg}(g)) \simeq g$. We set $c_g := \Psi_{Neg}(g)$. The definition of Φ yields variables $\alpha_1, \ldots \alpha_k$ and formulas $g', g_1, \ldots, g_k$ with $\bar{g} = \exists\alpha_1 \ldots \exists\alpha_k.(g' \wedge \neg g_1 \wedge \ldots \wedge \neg g_l)$.

When we build $\Psi(\exists\alpha.f')$, the existential step in the definition of Ψ is applied. We have to consider how this step modifies the graph $\Psi(f)$. We do an induction on $Neg_f \cup \{\top_f\}$, and we know that $(Neg_f \cup \{\top_f\}, \leq)$ and $(Cut \cup \{\top\}, \leq)$ are isomorphic. So we have to check how $area(c_g)$ is modified by the existential step. More precisely: We have to look at the vertices $v' \in V'$ with $\rho'(v') = *_\alpha$ and $ctx(v') = c_g$. It is sufficient to assume that we have only one vertex like this (if we have more vertices, we enumerate them and do the proof by induction). So let v' be the vertex with $\rho'(v') = *_\alpha$ and $ctx(v') = c_g$. There are two possibilities for v': Either v' is isolated, and we have $\kappa'(v') = C$ for a concept name $C \in \mathcal{C}$, or v' is incident with an edge and we have $\kappa'(v') = \top$. We distinguish these two cases.

We start with the first case, so let v'= $\boxed{C : *_\alpha}$ be a vertex with $ctx(v') = c_g$. During the existential step, v' is replaced by a vertex $\boxed{C : *}$, which is linked with an identity link to the new vertex v. When building inductively $\Phi(\mathfrak{G}, c)$ for the contexts $c \in Cut \cup \{\top\}$, we assign to v' a new variable β. So we see that we have

$$\begin{aligned}
&\Phi(\Psi(f), \Psi_{Neg}(g)) \\
&\quad = \quad \exists\alpha_1 \ldots . \exists\alpha_k.\exists\beta.(\alpha = \beta \wedge g'[\beta/\alpha] \wedge \neg g_1[\beta/\alpha] \wedge \ldots \wedge \neg g_l[\beta/\alpha]) \\
&\quad \overset{\text{Def.}\simeq}{\simeq} \quad \exists\alpha_1 \ldots \exists\alpha_k.(g' \wedge \neg g_1 \wedge \ldots \wedge \neg g_l) \\
&\quad = \quad \Phi(\Psi(f'), \Psi_{Neg}(g)) \\
&\quad \overset{\text{I.H. for } f'}{\simeq} \quad g
\end{aligned}$$

Hence the proof of the first case is completed.

Now we consider the second case, that is $v' = \boxed{\top : *_\alpha}$ and v' is incident with some edge. Then, during the existential step, v' is replaced by the new vertex v and erased afterwards. In $\Phi(\Psi(f'), \Psi_{Neg}(g))$, the vertex v' caused an atomic subformula $\top(\alpha)$ which does not occur in $\Phi(\Psi(f), \Psi_{Neg}(g))$. To put it more formally: We have a formula g'' with $g' = \top(\alpha) \wedge g''$ and $\Phi(\Psi(f), \Psi_{Neg}(g)) = \exists\alpha_1 \ldots \exists\alpha_k.(g' \wedge \neg g_1 \wedge \ldots \wedge \neg g_l)$. Especially we have

$$\Phi(\Psi(f), \Psi_{Neg}(g)) \overset{\text{Def.}\simeq}{\simeq} \Phi(\Psi(f'), \Psi_{Neg}(g)) \overset{\text{I.H. for } f'}{\simeq} g \quad ,$$

and the proof for the second case is completed, too.

For $f = \exists\alpha.f'$, we have shown that (11.1) holds for all $g \in Neg_f \cup \{\top_f\}$ with $g \neq f$. It remains to prove that (11.1) holds for $g = f$.

The new vertex v causes a new atomic subformula $\top(\alpha)$ of f. In other respects the proof of (11.1) for $g = f$ can be performed like the proof for $g < f$. As we have $(h \wedge \top(\alpha)) \simeq h$ for arbitrary formulas h, we conclude that $f \simeq \Phi(\Psi(f), \Psi_{Neg}(f))$ holds too.

For $g = f$, we have $\Phi(\Psi(f), \Psi_{Neg}(f)) = \Phi(\Psi(f), \top) = \Phi(\Psi(f))$. Thus (11.1) yields the lemma. □

The next theorem is an immediate consequence of the previous lemma.

Theorem 11.8 ($f \leftrightarrow \Phi(\Psi(f))$ for Formulas).
For each formula $f \in FOL$ we have $\vdash f \longleftrightarrow \Phi(\Psi(f))$. In particular we have $f \vdash \Phi(\Psi(f))$ and $\Phi(\Psi(f)) \vdash f$.

Proof: Let $f \in$ FOL be a formula. By renaming the bound variables in f, we transform f into a formula f' which is variable-purified. It is easy to see that we have $\Psi(f) = \Psi(f')$. We conclude

$$f \vdash f' \overset{\text{Lem. 11.7}}{\vdash} \Phi(\Psi(f')) = \Phi(\Psi(f)) \qquad \text{and}$$

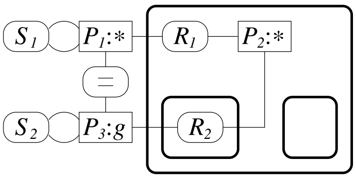
$$\Phi(\Psi(f)) = \Phi(\Psi(f')) \overset{\text{Lem. 11.7}}{\vdash} f' \vdash f$$

This yields the theorem. □

11.2 Equivalence of $\mathfrak{G}$ and $\Psi(\Phi(\mathfrak{G}))$ for Graphs $\mathfrak{G}$

This section is the counterpart of Sect. 11.1 for concept graphs. We will show that each concept graph $\mathfrak{G}$ is provably equivalent to $\Psi(\Phi(\mathfrak{G}))$ (i.e., we have $\mathfrak{G} \vdash \Psi(\Phi(\mathfrak{G}))$ and $\Psi(\Phi(\mathfrak{G})) \vdash \mathfrak{G}$). Again we provide an example before we start the formal proof. In this example, the idea of the proof is carried out. For understanding the proof, this example should be helpful.

Let $\mathfrak{G}$ be the following graph:



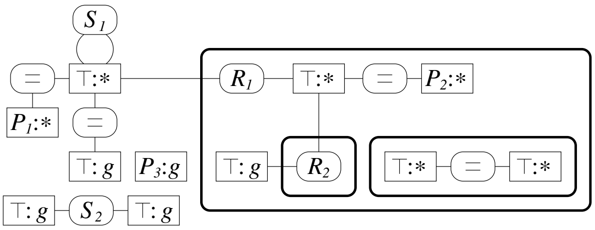
We want to emphasize that in this example, we tried to take various features of concept graphs into account which have to be captured by the proof, namely

- generic and non generic concept boxes which are incident with relations,
- different nestings of negations,
- an empty cut, and
- vertices which are multi-fold incident with the same edge.

The translation of $\mathfrak{G}$ to FOL is:

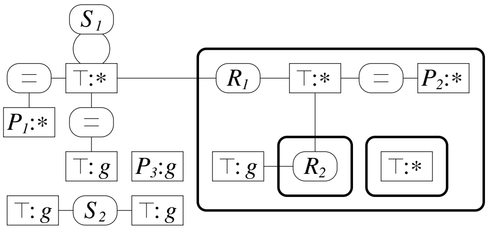
$$\begin{aligned}\Phi(\mathfrak{G}) = \exists x.P_1(x) \wedge P_3(g) \wedge x = g \wedge S_1(x,x) \wedge S_2(g,g)\\ \wedge\neg(\exists y.P_2(y) \wedge R_1(x,y) \wedge \neg(R_2(g,y) \wedge \neg\exists z.\top(z)))\end{aligned}$$

Thus $\mathfrak{G}' := \Psi(\Phi(\mathfrak{G}))$ is the graph on the right:

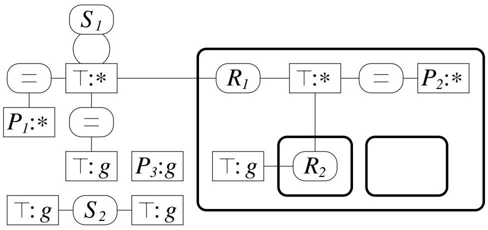


We want to show that we can derive $\mathfrak{G}$ from $\mathfrak{G}'$, and vice versa. This is done as follows:

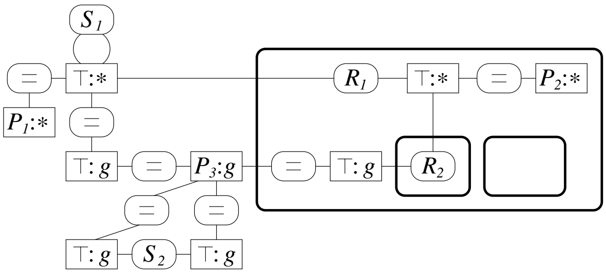
In $\mathfrak{G}'$, in the former empty cut, the two vertices are merged:



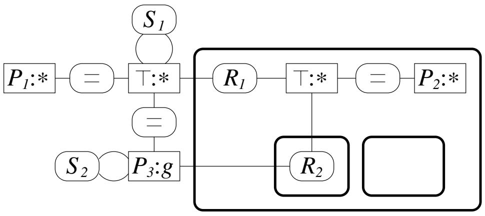
Using ⊤-erasure, the remaining vertex in the former empty cut is deleted (this graph will be called $\mathfrak{G}^{(2)}$ in the proof):



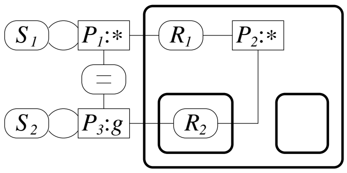
Now we insert a new identity link between $\boxed{P_3\colon g}$ and each vertex $\boxed{\top\colon g}$ (this graph will be called $\mathfrak{G}^{(3)}$ in the proof):



All vertices $\boxed{\top : g}$ are merged into $\boxed{P_3 : g}$:



Now we merge the vertices $\boxed{\top : *}$ into the vertices $\boxed{P_1 : *}$ resp. $\boxed{P_2 : *}$ (this graph will be called $\mathfrak{G}^{(4)}$ in the proof):



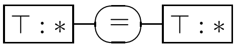
This is $\mathfrak{G}$ again. Note that all steps during the proof can be executed in both directions, hence we have $\mathfrak{G}' \vdash \mathfrak{G}$ and $\mathfrak{G} \vdash \mathfrak{G}'$.

We have just exemplified the procedure to derive $\Psi(\Phi(\mathfrak{G}))$ from $\mathfrak{G}$ and vice versa. The idea of this procedure will will be worked out in the following theorem.

Theorem 11.9 ($\mathfrak{G}$ and $\Psi(\Phi(\mathfrak{G}))$ are Equivalent).
For each concept graph $\mathfrak{G}$, we have

$$\mathfrak{G} \vdash \Psi(\Phi(\mathfrak{G})) \qquad \textit{and} \qquad \Psi(\Phi(\mathfrak{G})) \vdash \mathfrak{G} \quad .$$

Proof: We define $\mathfrak{G} := (V, E, \nu, \top, Cut, area, \kappa, \rho)$, $h := \Phi(\mathfrak{G})$ and $\mathfrak{G}' := \Psi(h) := (V', E', \nu', \top', Cut', area', \kappa', \rho')$. If $\mathfrak{G}$ is the empty graph, we have

$$\mathfrak{G}' := \Psi(\Phi(\mathfrak{G})) := \Psi(\exists x.\top(x)) :=$$  .

By using the $\top$-insertion rule and the iteration rule, this graph can be derived from the empty sheet of assertion. So we have $\mathfrak{G} \vdash \mathfrak{G}'$. This derivation can be performed in both directions, hence we have $\mathfrak{G}' \vdash \mathfrak{G}$ too. So the theorem holds for the empty graph, and, for the rest of the proof, we can assume that $\mathfrak{G}$ is not the empty graph.

First we note that the atomic subformulas[2] of h are as follows:

– For each $v \in V$, $\phi(v) := \kappa(v)(\Phi_t(v))$ is an atomic subformula of h,

– for each $e \in E$ with $e = (v_1, \ldots, v_n)$, $\phi(e) := \kappa(e)(\Phi_t(v_1), \ldots, \Phi_t(v_n))$ is an atomic subformula of h,

[2] A subformula may occur several times in G, hence we should more precisely speak of *occurences of subformulas*. We avoided this term to improve the readability of the proof, but have in mind that in fact we refer to occurences of subformulas instead of subformulas.

– for each context $c \in Cut \cup \{\top\}$ with $area(c) = \emptyset$, $\phi(c) := \top(\alpha_{empty})$ is an atomic subformula of h.

All these atomic subformulas are generated by vertices and edges of $\mathfrak{G}$. When we translate h back to a concept graph (i.e., when we build $\Psi(h) = \Psi(\Phi(\mathfrak{G}))$), these subformulas generate in turn vertices and edges of $\mathfrak{G}'$. But we want to stress that each application of the existential step in the definition of Ψ erases vertices $\boxed{\top : *_\alpha}$ which are generated by atomic subformulas $\phi(e), e \in E$, and adds a new vertex $\boxed{\top : *}$ as well as a new identity link.

We start with some simple observations. It is easy to that for $k \in V \cup E$ and each context $c \in Cut \cup \{\top\}$, we have

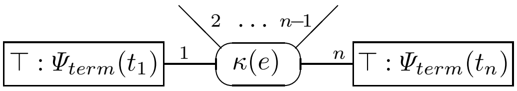
$$k < c \Longleftrightarrow \phi(k) \text{ is a subformula of } \Phi(\mathfrak{G}, c)\,. \tag{11.2}$$

Analogously we have for contexts $c_1, c_2 \in Cut \cup \{\top\}$

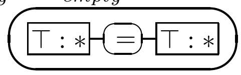
$$c_1 \leq c_2 \Longleftrightarrow \Phi(\mathfrak{G}, c_1) \text{ is a subformula of } \Phi(\mathfrak{G}, c_2)\,. \tag{11.3}$$

To each cut $c \in Cut$ belongs a subformula $\neg\Phi(\mathfrak{G}, c)$ of h. The negation sign at the beginning of this subformula is translated back into a cut $c' \in Cut'$. Hence we can set $f_{Cut}(c) := c'$. This defines a mapping $f_{Cut} : Cut \to Cut'$. It is easy to see that f_{Cut} is bijective.

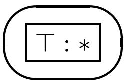
For each edge $e \in E$, the subformula $\phi(e)$ is translated back into an edge

$e' =$ 

We set $f_E(e) := e'$. It is easy to see that $f_E : E \to E'$ is injective. But f_E is not surjective. We have two kinds of further identity links in $\mathfrak{G}'$ which are not in the range of f_E: First we have identity links which are generated when the existential step is applied to a subformula $\phi(c) = \top(\alpha_{empty})$ where c is an empty cut of $\mathfrak{G}$. Second we have identity links which are generated when the existential step is applied to a variable $\alpha \neq \alpha_{empty}$ (i.e., a variable which is generated by a concept box $\boxed{P : *}$ in $\mathfrak{G}$). We have to transform $\mathfrak{G}'$ in order to delete this additional identity links.

Let us start with the identity links of the first kind, i.e., e start our derivation of $\mathfrak{G}' \vdash \mathfrak{G}$ with an investigation of the empty cuts $c \in Cut$. Each empty cut c generates a subformula $\neg\exists\alpha_{empty}.\alpha_{empty} = \alpha_{empty}$ of h. These subformulas are translated back into subgraphs $\mathfrak{G}'_c :=$  .

We now derive from $\mathfrak{G}'$ a new graph $\mathfrak{G}^{(2)}$ as follows: In each of the subgraphs $\mathfrak{G}_c$, we first merge on of the generic nodes into the other one (so we get



), and the remaining node is erased with the ⊤-erasure-rule. After we have done this for all empty cuts $c \in Cut$, we have derived from $\mathfrak{G}'$ the new graph $\mathfrak{G}^{(2)}$. The mappings $f_{Cut} : Cut \cup \{\top\} \to Cut' \cup \{\top'\}$ and $f_E : E \to E'$ can canonically be converted to mappings $f_{Cut} : Cut \cup \{\top\} \to Cut^{(2)} \cup \{\top^{(2)}\}$ and $f_E : E \to E^{(2)}$.

As we have erased all identity links which are generated by subformulas ϕ_c where $c \in Cut$ is an empty cut, we now have that $f_{Cut}(c)$ is empty. The two rules we have applied to derive $\mathfrak{G}^{(2)}$ (merging two vertices and ⊤-erasure) can be reversed, thus we have $\mathfrak{G}' \vdash \mathfrak{G}^{(2)}$ and $\mathfrak{G}^{(2)} \vdash \mathfrak{G}'$.

As we erased all identity links of the first kind, i.e., identity links which are generated by empty cuts in $\mathfrak{G}$, we will still have to erase identity links of the second kind, i.e., identity links which are generated when the existential step is applied to a variable $\alpha \neq \alpha_{empty}$. This will be a part of the remaining proof.

Now we consider the vertices and edges in $\mathfrak{G}'$ which are generated by vertices $v \in V$. Each v generates a subformula $\phi(v) = \kappa(v)(\varPhi_t(v))$ of h. In particular we have an occurrence $\varPhi_t(v)$ in the subformula $\phi(v)$ of h. We denote this occurrence by $occ(v)$. Furthermore, each edge $e \in E$ and each $i \in \mathbb{N}$ with $e|_i = v$ generates a subformula $\phi(e) := \kappa(e)(\ldots, \varPhi_t(v), \ldots)$, hence it generates an occurrence of $\varPhi_t(v)$ in h. The set of all these occurrences of $\varPhi_t(v)$ is denoted by $Occ(v)$.

To make this definition more transparent, we have a look at the example at the beginning of this section. Let v be the vertex $\boxed{P_1 : *}$. We have set $\varPhi_t(v) := x$. In $\varPhi(\mathfrak{G})$, the first underlined variable x is the occurrence $occ(v)$. The remaining variables x are the occurrences in $Occ(v)$.

$$\begin{aligned}\varPhi(\mathfrak{G}) = \exists x.P_1(\underline{x}) \wedge P_3(g) \wedge \underline{x} = g \wedge S_1(\underline{x}, \underline{x}) \wedge S_2(g, g) \\ \wedge\neg(\exists y.P_2(y) \wedge R_1(\underline{x}, y) \wedge \neg(R_2(g, y) \wedge \neg\exists z.\top(z))\end{aligned}$$

Now we distinguish between generic concept boxes $v \in V$ (i.e., $\rho(v) = *$) and non-generic concept boxes $v \in V$ (i.e., $\rho(v) \in \mathcal{G}$).

- We start with the first case, so let $v \in V$ be a generic concept box. Let $ctx(v) = c$. The occurrence $occ(v)$ is an occurrence in a subformula $\kappa(v)(occ(v))$ of $\varPhi(\mathfrak{G}, c)$. This subformula is atomic and therefore – at the beginning of the iterative construction of $\varPsi(h)$ – translated to the isolated concept box $\boxed{\kappa(v) : \varPhi_t(v)}$. Furthermore, each edge $e \in E$ with $e|_i = v$ generates an occurrence $j \in Occ(v)$ which is an occurrence in a subformula $\phi(e) = \kappa(e)(\ldots, j, \ldots)$. These occurrences are translated to concept boxes $\boxed{\top : \varPhi_t(v)}$. Note that $\varPhi(\mathfrak{G}, c)$ starts with some existential quantifications, namely a quantification $\exists\varPhi_t(v)$. So, when building $\varPsi(\varPhi(\mathfrak{G}, c))$, the existential step is applied: A new concept box $v' = \boxed{\top : *}$ is inserted into $area(c)$, each concept box $\boxed{\kappa(v) : \Phi_t(v)}$ is substituted by v', and the box $\boxed{\kappa(v) : \Phi_t(v)}$ is replaced by $\boxed{\kappa(v) : *}$ and linked with an identity link to v'. Thus, after this step, we have two concept boxes $\boxed{\kappa(v) : *}$ and $\boxed{\top : *}$ which are placed in $area(v)$ and linked with an identity link. During the further iterative construction of $\Psi(h)$, these nodes are not changed anymore. During the derivation of $\mathfrak{G}^{(2)}$ from $\mathfrak{G}'$, these nodes were not changed, neither. So we can define mappings $f_V(v) : V^* \to V^{(2)}$ and $\bar{f}_V(v) : V^* \to \mathfrak{P}(V^{(2)})$ which map to these two concept boxes as follows:[3]

$$f_V(v) := \boxed{\kappa(v) : *} \quad \text{and} \quad \bar{f}_V(v) := \{ \boxed{\top : *} \}$$

Furthermore we have

$$\begin{aligned} e|_i = v \text{ in } \mathfrak{G} &\Longleftrightarrow \kappa(e)(\ldots, \Phi_t(v), \ldots) \text{ is a subformula of } h \\ &\Longleftrightarrow f_E(e)|_i \in \bar{f}_V(v) \end{aligned} \tag{11.4}$$

This condition will be used later in the proof.

- Now let $v \in V$ be a concept box with $\rho(v) \in \mathcal{G}$, i.e., $v = \boxed{\kappa(v) : \rho(v)}$. The box v is translated to a subformula $\kappa(v)(\rho(v))$ of h. The object name $\rho(v)$ in this subformula is the occurrence $occ(v)$. The subformula is translated back to a concept box $v' = \boxed{\kappa(v) : \rho(v)}$ in $\mathfrak{G}'$. We set $f_V(v) := v'$.

  Again each occurrence $j \in Occ(v)$ is an occurrence in a subformula $\phi(e) = \kappa(e)(\ldots, j, \ldots)$ of $\Phi(\mathfrak{G}, c)$. All these occurrences are translated to concept boxes $\boxed{\top : \rho(v)}$. Let $\bar{f}_V(v)$ be the set of all these concept boxes. A similar argumentation like in the first case yields that the range of these mappings is a subset of $V^{(2)}$, i.e., we have $f_V(v) : V^{\mathcal{G}} \to V^{(2)}$ and $\bar{f}_V(v) : V^{\mathcal{G}} \to \mathfrak{P}(V^{(2)})$.

  Again we have

$$\begin{aligned} e|_i = v \text{ in } \mathfrak{G} &\Longleftrightarrow \kappa(e)(\ldots, \rho(v), \ldots) \text{ is a subformula of } h \\ &\Longleftrightarrow f_E(e)|_i \in \bar{f}_V(v) \end{aligned} \tag{11.5}$$

Both cases together yield mappings $f_V(v) : V \to V^{(2)}$ and $\bar{f}_V(v) : V \to \mathfrak{P}(V^{(2)})$. We have furthermore

$$V^{(2)} = \dot{\bigcup}_{v \in V} (\{f_V(v)\} \mathbin{\dot{\cup}} \bar{f}_V(v)) \tag{11.6}$$

Now, for each $v \in V^{\mathcal{G}}$, the following is done in $\mathfrak{G}^{(2)}$: For each $v' \in \bar{f}_V(v)$, an identity link is inserted between $f_V(v)$ and v'. We get a new graph $\mathfrak{G}^{(3)}$ which

[3] It will become clear in the case $\rho(v) \in \mathcal{G}$ why we define $\bar{f}_V(v)$ as a set.

is provably equivalent to $\mathfrak{G}^{(2)}$. The mappings $f_V : V \to V^{(2)}$, $f_E : E \to E^{(2)}$ and $f_{Cut} : Cut \cup \{\top\} \to Cut^{(2)} \cup \{\top^{(2)}\}$ can canonically be transformed to mappings $f_V : V \to V^{(3)}$, $f_E : E \to E^{(3)}$ and $f_{Cut} : Cut \cup \{\top\} \to Cut^{(3)} \cup \{\top^{(3)}\}$.

In $\mathfrak{G}^{(3)}$, we have for each $v \in V$ and each $v' \in \bar{f}_V(v)$ an identity link between $f_V(v)$ and v'. Now, for each $v \in V$ and each $v' \in \bar{f}_V(v)$, we merge v' into $f_V(v)$. So we derive a new graph $\mathfrak{G}^{(4)}$ which is provably equivalent to $\mathfrak{G}^{(3)}$. Again the mappings $f_V : V \to V^{(3)}$, $f_E : E \to E^{(3)}$ and $f_{Cut} : Cut \cup \{\top\} \to Cut^{(3)} \cup \{\top^{(3)}\}$ can canonically be transformed to mappings $f_V : V \to V^{(4)}$, $f_E : E \to E^{(4)}$ and $f_{Cut} : Cut \cup \{\top\} \to Cut^{(4)} \cup \{\top^{(4)}\}$. We have erased all vertices $v' \in \bigcup_{v \in V} \bar{f}_V(v)$, thus the mapping $f_V : V \to V^{(4)}$ is bijective. As we have for each $v \in V$ erased all identity links between between $f_V(v)$ and each $v' \in \bar{f}_V(v)$, the mapping $f_E : E \to E^{(4)}$ is bijective. The conditions (11.4) and (11.5) yield that we have

$$e|_i = v \Longrightarrow f_E(e)|_i = f_V(v) \text{ for each } v \in V\,.$$

It is easy to see that $\rho(v) = \rho^{(4)}(f_V(v))$ and $\kappa(v) = \kappa^{(4)}(f_V(v))$ hold for all $v \in V$, and $\kappa(e) = \kappa^{(4)}(f_E(e))$ holds for all $e \in E$. Finally, we conclude from (11.2) and (11.3) that we have

$$f[area(c)] = area^{(4)}(f(c)) \text{ for each } c \in Cut \cup \{\top\}\,.$$

Hence $f := f_V \cup f_E \cup f_{Cut}$ is an isomorphism from $\mathfrak{G}$ to $\mathfrak{G}^{(4)}$. Furthermore we have $\mathfrak{G}^{(4)} \vdash \Psi(\Phi(\mathfrak{G}))$ and $\Psi(\Phi(\mathfrak{G})) \vdash \mathfrak{G}^{(4)}$, so the proof is finished. □

11.3 Ψ Respects $\vdash$

In this section we want to show that Ψ respects the derivability relation $\vdash$, i.e., we want to show that we have $f_1 \vdash f_2 \Longrightarrow \Psi(f_1) \vdash \Psi(f_2)$ for formulas f_1, f_2. The relations $\vdash$ for formulas resp. for concept graphs are based on the appropriate calculi, so the idea of the proof is to show that Ψ respects every rule of the calculus for FOL. We want to point out that the calculus for FOL is based on formulas with free variables, in contrast to the calculus on CG which is based on concept graphs without variables. For this reason we have to translate formulas with free variables to concept graphs without variables. This can be done in two canonically given ways and is captured by the following definition.

Definition 11.10 (Existential and Universal Closures of Formulas). *Let f be a formula with $Free(f) = \alpha_1, \ldots, \alpha_n$. Then $f_\exists := \exists\alpha_1 \ldots \exists\alpha_n f$ is called* existential closure of f *and $f_\forall := \neg\exists\alpha_1 \ldots \exists\alpha_n \neg f$ is called* universal closure of f. *Now we set*

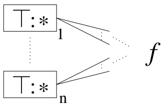
$$\Psi_\exists := \begin{cases} FOL \to CG \\ \quad f \mapsto \Psi(f_\exists) \end{cases} \qquad \textit{and} \qquad \Psi_\forall := \begin{cases} FOL \to CG \\ \quad f \mapsto \Psi(f_\forall) \end{cases}$$

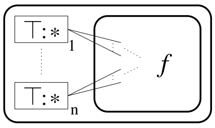
For each formula, $\Psi_\exists(f)$ and $\Psi_\forall(f)$ are concept graphs without variables. The definition of the relation $\models$ in FOL yields that a formula f is valid in a relational structure if and only if $f_\forall$ is valid in that structure. For this reason we have to focus on the universal closure of f and hence on the mapping $\Psi_\forall$.

If $\Psi(f)$ is the translation of f into a concept graph, then (informally spoken) $\Psi_\forall(f)$ is build from $\Psi(f)$ as follows:

1. First a new cut is drawn around $\Psi(f)$.
2. For each free variable α_i a new concept box $v_i :=\boxed{\top : *}$ is drawn on the sheet of assertion. Each concept box $\boxed{\top : *_{\alpha_i}}$ which is incident with an edge is substituted by v_i. From each isolated concept box $\boxed{P : *_{\alpha_i}}$, an identity link is drawn to v_i and, in $\boxed{P : *_{\alpha_i}}$, the indexed generic marker $*_\alpha$ is replaced by $*$.
3. Finally, a new cut is drawn around the whole graph (in particular around the new vertices v_i).

The mappings $\Psi_\exists$ and $\Psi_\forall$ shall therefore be (informally) sketched as follows:[4]

$\Psi_\exists(f) :=$ 

and $\Psi_\forall(f) :=$ 

Now we are prepared to prove that Ψ respects $\vdash$.

Lemma 11.11 (Ψ Respects Syntactical Entailment).
Let $f_1, \ldots, f_n \vdash g$, $n \in \mathbb{N}_0$ be a rule of the calculus for FOL. Then we have $\Psi_\forall(f_1), \ldots, \Psi_\forall(f_n) \vdash \Psi_\forall(g)$ in the calculus for concept graphs.

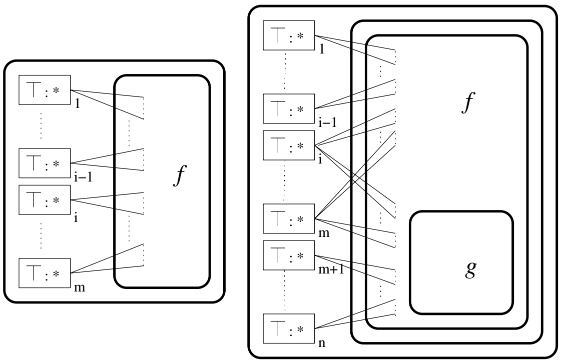
Proof: We have to show the lemma for each rule.

Modus Ponens: $f, f \to g \quad \vdash \quad g$

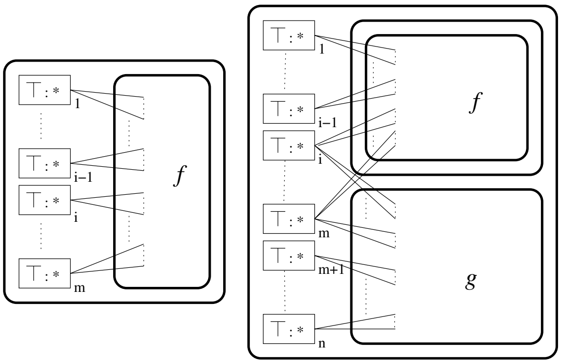
Let f, g be two formulas. Without loss of generality let Free$(f) = \{x_1, \ldots x_m\}$, and Free$(g) = \{x_i, \ldots x_n\}$.

[4] Of course f shall stand for the subgraph which is generated by f. But this generated subgraph is *not* $\Psi(f)$, but the subgraph we get after the existential steps are applied to $\Psi(f)$. For this reason we write f instead $\Psi(f)$ in this diagrams, although f shall denote a subgraph of the written graph.

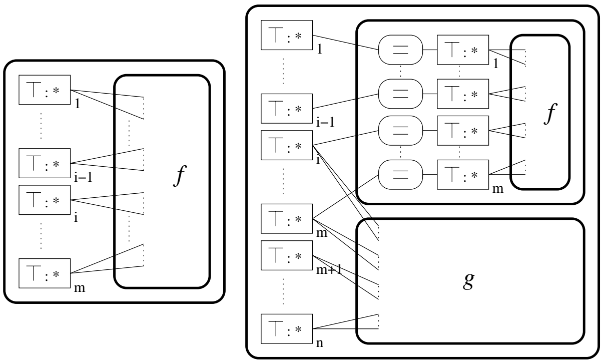
The graph we start with is the juxtaposition of the graphs $\Psi_\forall(f)$, $\Psi_\forall(f \to g)$:



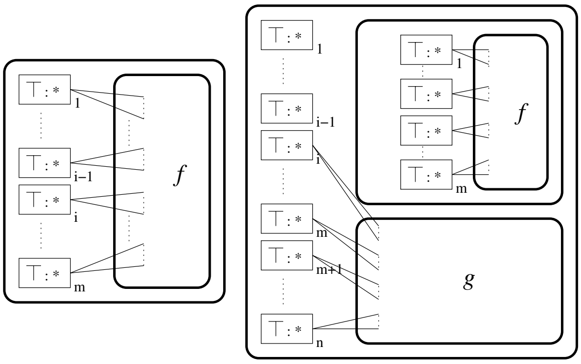
The double cut rule yields:



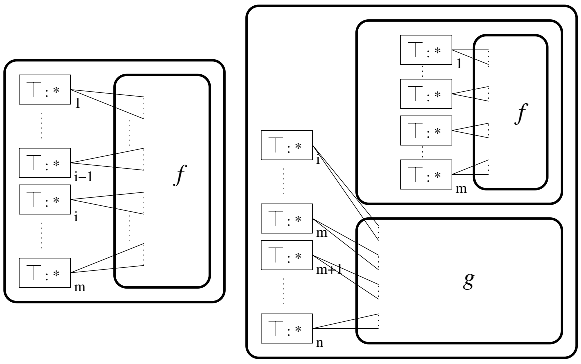
We split each vertex $\boxed{\top : *}$ which has an index from 1 to $i-1$:



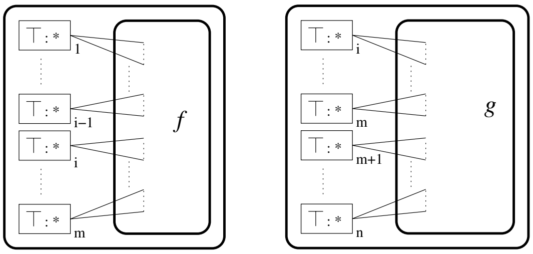
All new identity links are erased:



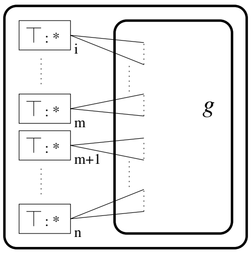
Using ⊤-erasure, all vertices $\boxed{\top : *}$ with an index between 1 and $i-1$ are now erased:



Deiteration yields:



Now erasure yields:

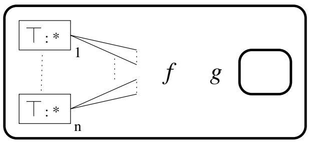


This is $\Psi_\forall(g)$, hence we have $\Psi_\forall(f)\,,\, \Psi_\forall(f \to g) \vdash \Psi_\forall(g)$.

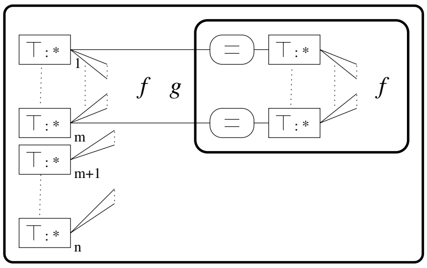
P1: $\vdash\ f \to (g \to f)$

Let f, g be two formulas. Without loss of generality let $\mathrm{Free}(f) := \{x_1, \ldots x_m\}$ and $\mathrm{Free}(g) := \{x_i, \ldots x_n\}$.

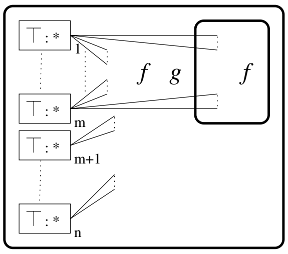
From the sheet of assertion, we can derive with double cut and insertion:



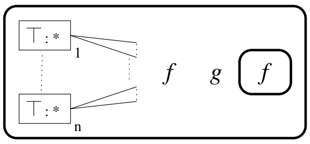
Iteration of the subgraph which is generated by f yields:



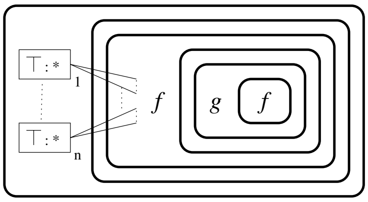
We merge the vertices $\boxed{\top : *}$ in the inner cut:



A shorter representation for this is:



Double cut yields:

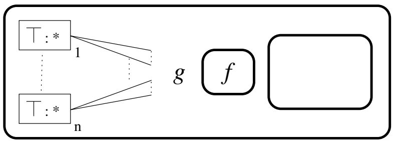


This is $\Psi_\forall((f \to (g \to f)))$.

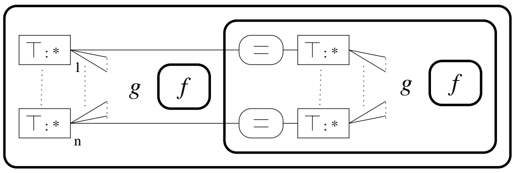
P2: $\vdash\ (\neg f \to \neg g) \to (g \to f)$

The scheme for the proof of P2 is the same like in the proof for P1. Again let f, g be two formulas with $\mathrm{Free}(f) := \{x_1, \ldots x_m\}$ and $\mathrm{Free}(g) := \{x_i, \ldots x_n\}$.

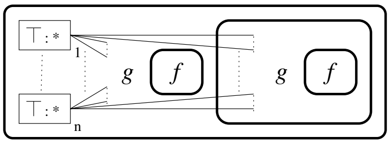
From the sheet of assertion, we can derive with double cut and insertion:



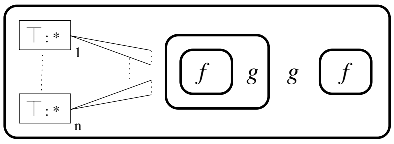
We iterate the inserted subgraph:



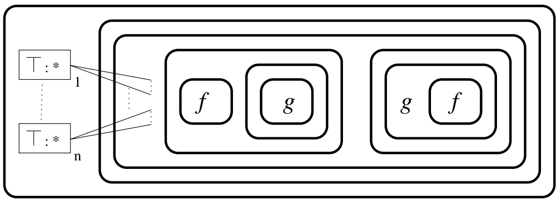
Now identity-erasure yields:



A shorter representation for this is:



Double cut yields:

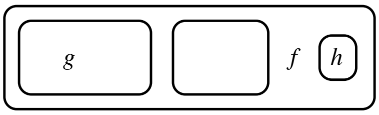


This is $\Psi_\forall((\neg f \to \neg g) \to (g \to f))$.

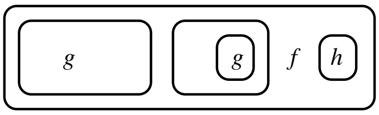
P3: $\vdash (f \to (g \to h)) \to ((f \to g) \to (f \to h))$

We have to show $\vdash \Psi_\forall((f \to (g \to h)) \to ((f \to g) \to (f \to h)))$. The scheme for the proof of P3 is the same like of the proofs for P1 and P2. From the empty sheet of assertion, we apply the rules double cut, insertion, iteration, identity-erasure to get the desired graph. To simplify matters we assume that f, g and h have no free variables (in case of free variables, an additional application of the identity-erasure-rule is needed). The proof is now done as follows:

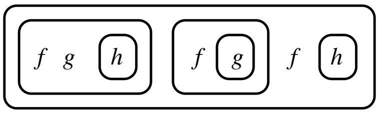
On the empty sheet of assertion we draw a double cut and insert f, $\boxed{g}$ and $\boxed{h}$ into the outer cut:



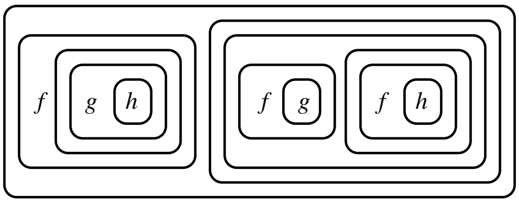
We iterate $\boxed{g}$ into the inner cut:



Now a twofold iteration of f and a iteration of $\boxed{h}$ yields:



Now a threefold application of the double cut rule yields:



This is $\Psi_\forall((f \to (g \to h)) \to ((f \to g) \to (f \to h)))$.

The rules MP, P1, P2 and P3 form a complete set of rules for propositional logic, hence they are in FOL sufficient to derive all tautologous formulas f. Hence we now have $\Psi_\forall(f)$ for all tautologous formulas f. This will be used in the rest of the proof.

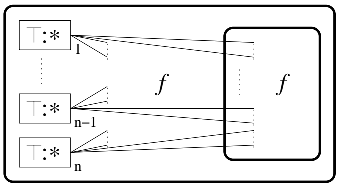
Ex1: $\vdash\ f \to \exists\alpha.f$

Let α be the variable x_n. We have to show $\vdash \Psi_\forall(f \to \exists x_n.f)$. To do this, we distinguish two cases.

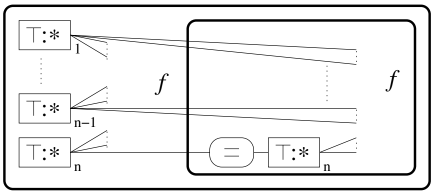
First, let $x_n \notin \mathit{Free}(f)$. Then we have $\Psi_\forall(f \to \exists x_n.f) = \Psi_\forall(f \to f)$. As $f \to f$ is tautologous, we can derive $\Psi_\forall(f \to f)$, i.e., we can derive the graph $\Psi_\forall(f \to \exists x_n.f)$.

Now let $x_n \in \mathit{Free}(f)$. Then $\Psi_\forall(f \to \exists x_n.f)$ is derived as follows:

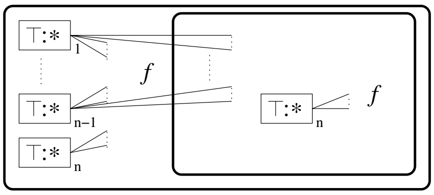
As $f \to f$ is tautologous, we can derive



The vertex $\boxed{\top : *}_n$ is splitted:



Now the new identity link is erased:



This is $\Psi_\forall(f \to \exists x_n.f)$.

Ex2: $\vdash\ f \to \exists\alpha_1.f[\alpha_1/\alpha_2]$, if $\alpha_1 \notin Free(f)$

Let f be an formula. We want to show $\vdash \Psi_\forall(f \to \exists\alpha_1.f[\alpha_1/\alpha_2])$, if $\alpha_1 \notin Free(f)$.

First we consider the case $\alpha_2 \notin Free(f)$. Then we have $f \to \exists\alpha_1.f[\alpha_2/\alpha_1] = f \to \exists\alpha_1.f$, and furthermore we have $\Psi_\forall(f \to \exists\alpha_1.f) = \Psi_\forall(f \to f)$. As $f \to f$ is tautologous, we conclude $\vdash \Psi_\forall(f \to f)$, thus $\vdash \Psi_\forall(f \to \exists\alpha_1.f[\alpha_2/\alpha_1])$.

Now let $\alpha_2 \in Free(f)$. It is easy to see that $\Psi(\exists\alpha_2.f) = \Psi(\exists\alpha_1.f[\alpha_2/\alpha_1])$ holds, thus we have $\Psi_\forall(f \to \exists\alpha_1.f[\alpha_2/\alpha_1]) = \Psi_\forall(f \to \exists\alpha_2.f)$. So this case can be reduced to the proof of Ex1.

Ex3: $\vdash\ f[\alpha/c] \to \exists\alpha.f$ for $c \in \mathcal{G}$

We have to show $\Psi_\forall(f[c/\alpha] \to \exists\alpha.f)$. Again we distinguish two cases.

First, let $\alpha \notin Free(f)$. Then we have $f[\alpha/c] = f$ and therefore

$$\Psi_\forall(f[\alpha/c] \to \exists\alpha.f) = \Psi_\forall(f \to \exists\alpha.f) = \Psi_\forall(f \to f)$$

Like in Ex1 or Ex2, we conclude $\vdash \Psi_\forall(f[\alpha/c] \to \exists\alpha.f)$.

Now let $\alpha \in Free(f)$. We will describe the general procedure how the graph $\Psi_\forall(f[\alpha/c] \to \exists\alpha.f)$ is derived. As this cannot be sketched as easy like in the proofs for Ex1 or Ex2, we exemplify this procedure with the formula

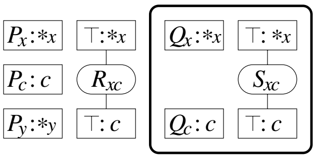
$$h = P_x(x) \wedge P_y(y) \wedge P_c(c) \wedge R_{xc}(x,c) \wedge \neg(Q_x(x) \wedge Q_c(c) \wedge S_{xc}(x,c))$$

with $P_x, P_y, P_c, Q_x, Q_c \in \mathcal{C}$ and $R_{xc}, S_{xc} \in \mathcal{R}$. We set $\alpha := x$ for this formula. This sample-formula takes various features of formulas into account which have to be captured by the proof, namely free variables (here: y) and arbitrary combinations of x, other variables and concept names as arguments in concept- and relation-names. In order to ease the understanding of the proof, the concept names and relation names are labelled with the terms they are used with.

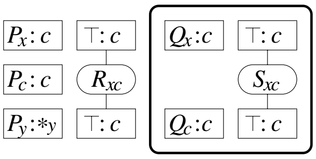
We want to show $\Psi_{\forall}(f[c/\alpha] \to \exists\alpha.f)$.

For our example h, we have:

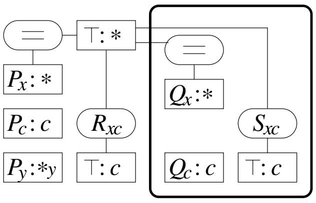
$\Psi(h) \;=$



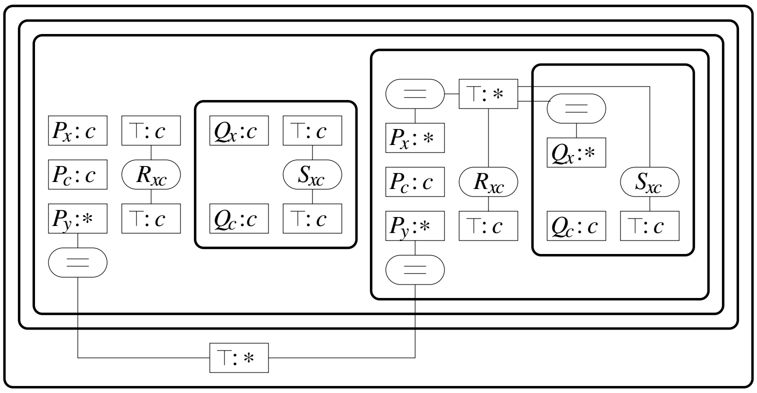
$\Psi(h[c/x] \;=$



$\Psi(\exists x.h) \;=$

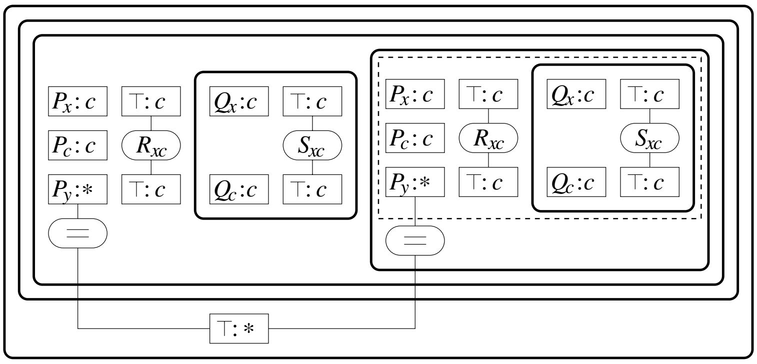


So $\Psi_{\forall}(h[c/x] \to \exists x.h)$ is the following graph:



To give a proof for the graph $\Psi_\forall(f[c/\alpha] \to \exists\alpha.f)$, we start with the formula $f[c/\alpha] \to f[c/\alpha]$. As this formula is a tautology, the graph $\Psi_\forall(f[c/\alpha] \to f[c/\alpha])$ can be derived.

For h this is the following graph (the marked area will be explained below):



The graph $\Psi(f[c/\alpha])$ generates two times a subgraph of $\Psi_\forall(f[c/\alpha] \to f[c/\alpha])$ (as premise as well as conclusion of the implication). We denote the subgraph which is the conclusion by $\mathfrak{G}_c$.

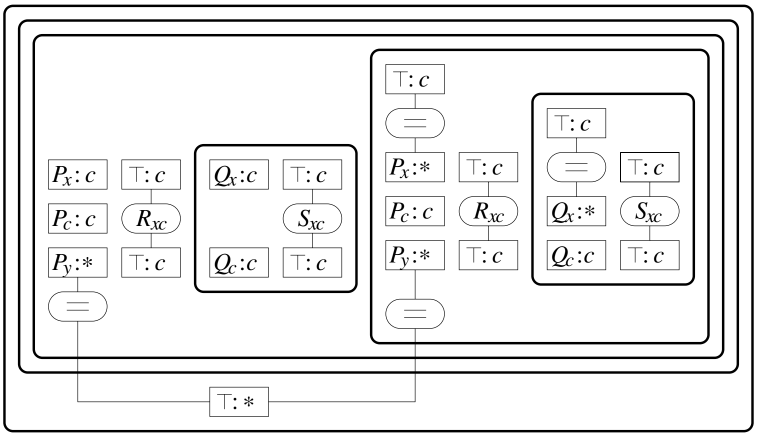
In our example, i.e., in the graph above, $\mathfrak{G}_c$ is marked subgraph.

In $\Psi_\forall(f[c/\alpha] \to f[c/\alpha])$ let V_1 be the set of all vertices $\boxed{P : c}$ which are generated by Ψ when a subformula $P[c/\alpha]$ (with $P \in \mathcal{C}$) of $f[c/\alpha]$ is translated. Furthermore let V_2 be the set of all vertices $\boxed{\top : c}$ which are generated by Ψ when a subformula $R(t_1, \ldots, t_n)[c/\alpha]$ (with $R \in \mathcal{R}_n$ and $\alpha \in \{t_1, \ldots, t_n\}$) of $f[c/\alpha]$ is translated.

In our example, V_1 consists of the two vertices $\boxed{P_x : c}$ and $\boxed{Q_x : c}$. The vertices $\boxed{P_c : c}$ and $\boxed{Q_c : c}$ do *not* belong to V_1, because in the corresponding subformulas $P_c(c)$ and $Q_c(c)$ of f, the terms c do not come into play by substituting x with c. Furthermore, V_2 consists of the two vertices $\boxed{\top : c}$ in the upper half of $\mathfrak{G}_c$.

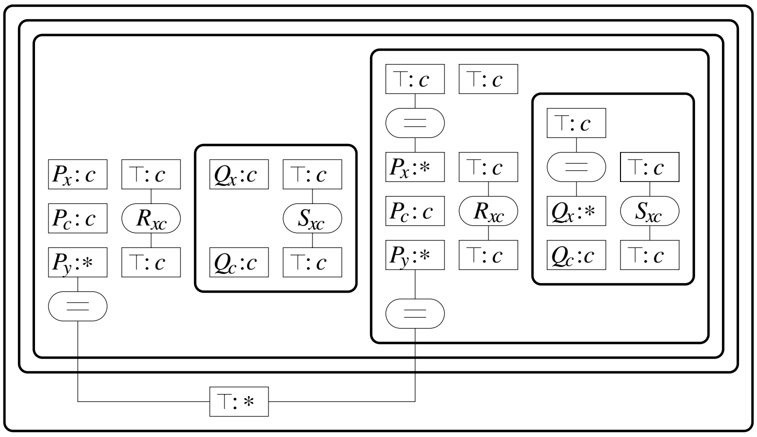
Now we use the congruence-rule (Lem. 10.3) to replace every vertex $\boxed{P : c}$ $\in V_1$ by $\boxed{\top : c}\!-\!(=)\!-\!\boxed{P : *}$.

For our example, this yields the following graph:



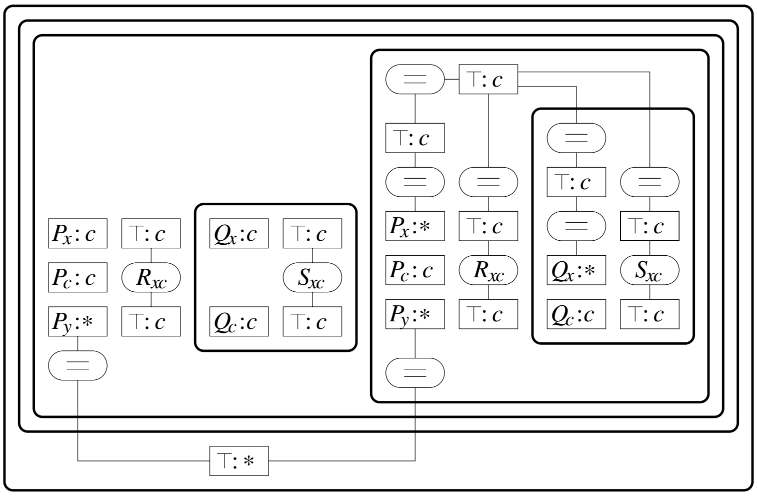
Into $\mathfrak{G}_c$ (hence into an even cut), a vertex $v_0 := \boxed{\top : c}$ is inserted by ⊤-insertion.

For our example, we get:



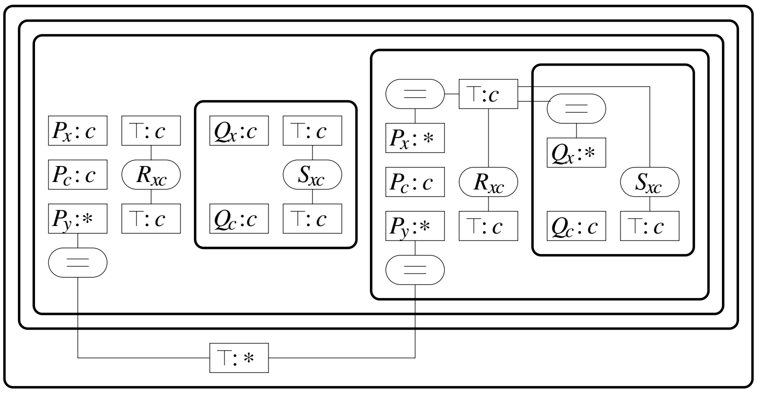
Using identity-insertion, we insert an identity link between v_0 and each vertex $\boxed{\top : c} \in V_1 \cup V_2$.

For our example, this yields the following graph:



Using $\top$-erasure, every vertex $\boxed{\top : c} \in V_1 \cup V_2$ is deleted.

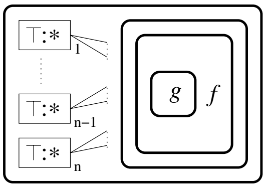
For our example, we get:

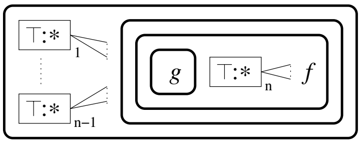


Finally we generalize the reference of v_0 to $*$, i.e., v_0 becomes $\boxed{\top : *}$. This yields $\Psi_\forall(f[c/\alpha] \to \exists\alpha.f)$, and the proof for Ex3 is finished.

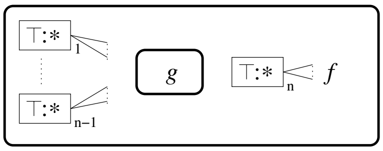
Ex4: $f \to g \vdash \exists\alpha.f \to g$, if $\alpha \notin Free(g)$

Let α be the variable x_n, and without loss of generality let $Free(f \to g) := \{x_1, \ldots x_n\}$. Suppose we have $\vdash \Psi_\forall(f \to g)$ with $x_n \notin Free(g)$. We have to derive the graph $\Psi_\forall(\exists x_n.f \to g)$. The graphs we have to consider are:

$\Psi_\forall(f \to g) \; =$  and

$\Psi_\forall(\exists x_n . f \to g) \; =$  .

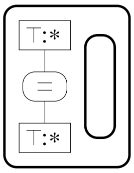
Now $x_n \notin \mathit{Free}(g)$ yields that we have no identity link between $\boxed{\top:*}_n$ and any vertex of . Hence, using double cut, both graphs are equivalent to:[5]



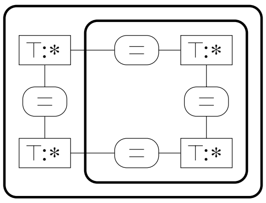
This yields that both graphs are equivalent, too, and to proof for Ex4 is finished.

Id1 (reflexivity of identity): $\vdash \alpha_0 = \alpha_0$

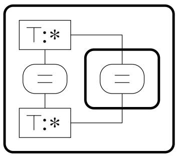
Double cut and insertion yield:



Iteration yields:

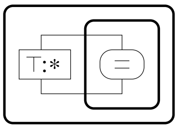


Now we merge the vertices in the inner cut into their counterparts in the outer cut:



[5] In fact $\forall x_n.(f \to g)$ and $\exists x_n.f \to g$ are equivalent, too.

The remaining vertices are merged:

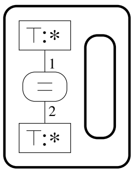


This is $\Psi_\forall(\alpha_0 = \alpha_0)$.

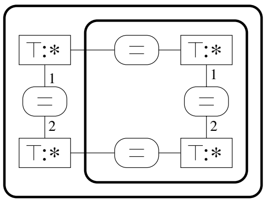
Id2 (symmetry of identity): $\vdash\ \alpha_0 = \alpha_1 \rightarrow \alpha_1 = \alpha_0$

In Lem. 10.5 we have already shown that identity links are symmetric, which will be used here. In the proof, we label the relevant identity links.

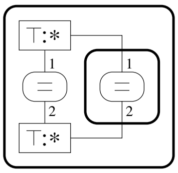
Again, we start with double cut and insertion:



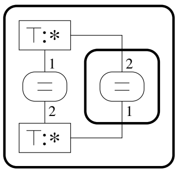
The iteration-rule is applied. The identity links are labelled as it is sketched in the graph.



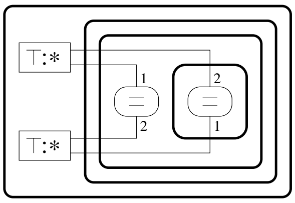
Now we merge the vertices in the inner cut into their counterparts in the outer cut:



Now we apply Lem. 10.5:



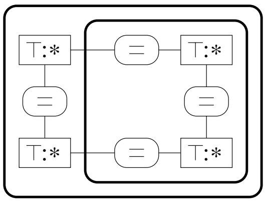
Double cut yields:



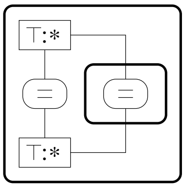
This is $\Psi_\forall(\alpha_0 = \alpha_1 \rightarrow \alpha_1 = \alpha_0)$.

Id3 (transitivity of identity): $\vdash \alpha_0 = \alpha_1 \to \alpha_1 = \alpha_2 \to \alpha_0 = \alpha_2$

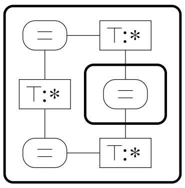
Similar to the proofs of Id1 and Id2, we derive from the sheet of assertion:



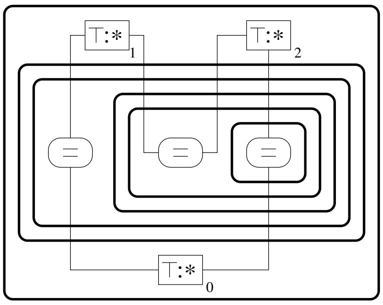
Now we merge the vertices in the inner cut into their counterparts in the outer cut:



Now one of the vertices is splitted:



We add two double cuts:

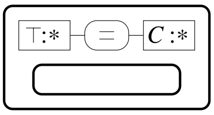


This is $\Psi_\forall(\alpha_0 = \alpha_1 \to \alpha_1 = \alpha_2 \to \alpha_0 = \alpha_3)$ (we have labelled the vertices so that it is easier to realize which vertex belongs to which variable).

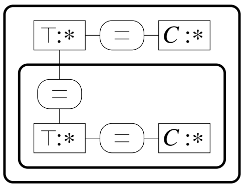
Cong1: $\vdash\ \alpha_0 = \alpha_1 \to C(\alpha_0) \to C(\alpha_1)$ for $C \in \mathcal{C}$

Let $C \in \mathcal{C}$. We have to derive $\Psi_\forall(\alpha_0 = \alpha_1 \to C(\alpha_0) \to C(\alpha_1))$

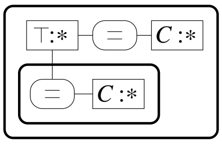
Double cut and insertion yield:



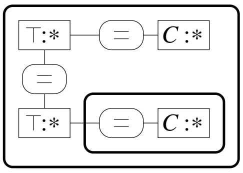
Iteration yields (note that one of the two vertices is linked to its copy):



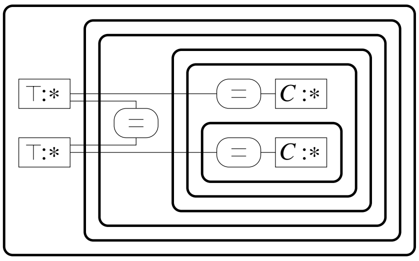
Now we merge the vertex in the inner cut into its counterpart in the outer cut:



The remaining vertex is splitted:



We add two double cuts:

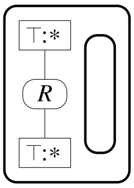


This is $\Psi_\forall(\alpha_0 = \alpha_1 \to P(\alpha_0) \to P(\alpha_1))$.

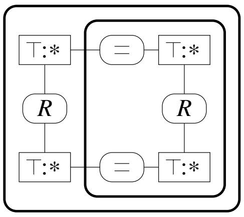
Cong2: $\vdash \alpha_0 = \alpha_n \to \ldots \to \alpha_{n-1} = \alpha_{2n-1} \to R(\alpha_0, \ldots, \alpha_{n-1}) \to R(\alpha_n, \ldots, \alpha_{2n})$

We perform this proof only for relation names with the arity 2. This proof can be performed as well for higher (or lower) arities, but then the graphs we need have a poor readability. So let $R \in \mathcal{R}_2$. We have to derive the concept graph $\Psi_\forall(\alpha_0 = \alpha_2 \to \alpha_1 = \alpha_3 \to R(\alpha_0, \alpha_1) \to R(\alpha_2, \alpha_3))$.

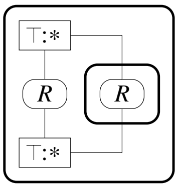
Double cut and insertion yield:



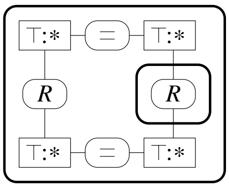
Iteration yields:



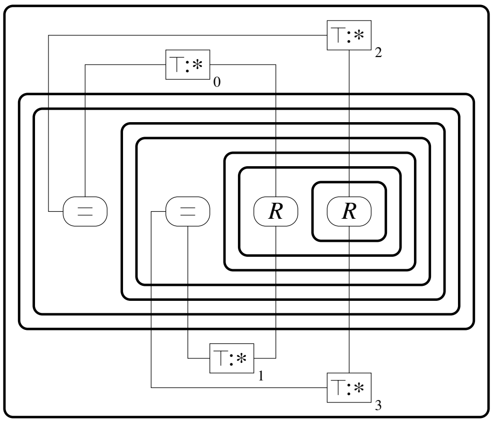
Now we merge the vertices in the inner cut into their counterparts in the outer cut:



Both remaining vertices are splitted:



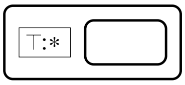
We add three double cuts:



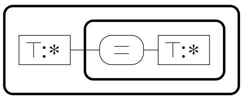
This is $\alpha_0 = \alpha_2 \to \alpha_1 = \alpha_3 \to R(\alpha_0, \alpha_1) \to R(\alpha_2, \alpha_3)$ (again we have labelled the vertices, so that it is easier to realize which vertex belongs to which variable).

Ax1: $\vdash \top(\alpha)$

Double cut and insertion yield:



Iteration yields:

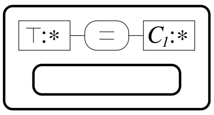


This is $\Psi_\forall(\top(\alpha))$.

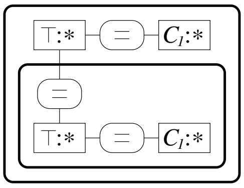
Ax2: $\vdash C_1(\alpha) \to C_2(\alpha)$ for $C_1, C_2 \in \mathcal{C}$ with $C_1 \leq_{\mathcal{C}} C_2$

Let $C_1, C_2 \in \mathcal{C}$ with $C_1 \leq C_2$. We have to derive $C_1(\alpha) \to C_2(\alpha)$.

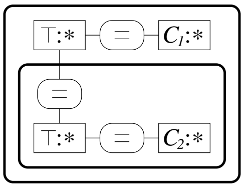
Double cut and insertion yield:



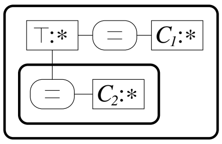
Iteration yields (note that one of the two vertices is linked to its copy):



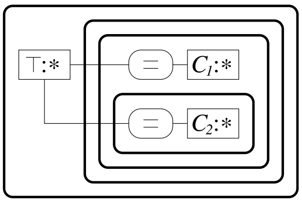
In the inner vertex $\boxed{C_1{:}*}$, we generalize the concept name:



We merge the vertex $\boxed{\top{:}*}$ in the inner cut into its counterpart in the outer cut:



Double cut yields:

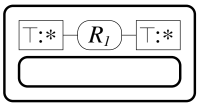


This is $\Psi_\forall(C_1(\alpha) \to C_2(\alpha))$.

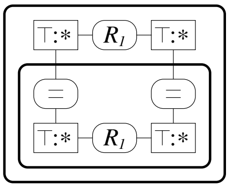
Ax3: $\vdash R_1(\alpha_1, \dots, \alpha_n) \to R_2(\alpha_1, \dots, \alpha_n)$ for $R_1, R_2 \in \mathcal{R}_k$ with $R_1 \leq_{\mathcal{R}} R_2$

For the same reasons as in Cong2, we perform this proof only for relation names with the arity 2. So let $R_1, R_2 \in \mathcal{R}_2$ with $R_1 \leq R_2$. We have to derive $R_1(\alpha_1, \alpha_2) \to R_2(\alpha_1, \alpha_2)$.

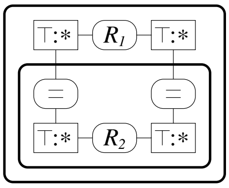
Double cut and insertion yield:



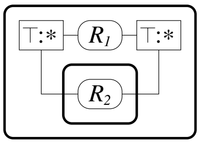
Iteration yields:



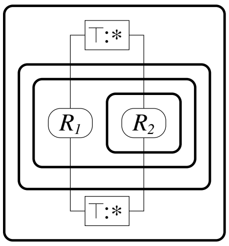
We generalize the relation name of the inner edge:



Now we merge both vertices in the inner cut into their counterparts in the outer cut:



Double cut yields:



This is $\Psi_\forall(R_1(\alpha_1, \alpha_2) \to R_2(\alpha_1, \alpha_2))$. □

We have shown that each axiom and rule of the FOL-calculus is respected by the mapping $\Psi : \text{FOL} \to \text{CG}$. Thus the last lemma yields immediately that $\Psi : \text{FOL} \to \text{CG}$ respects the syntactical entailment relation $\vdash$, i.e., we have the following theorem.

Theorem 11.12 (Main Syntactical Theorem for Ψ).
Let f_1, f_2 be two FOL-formulas (possibly with free variables). Then we have

$$f_1 \vdash f_2 \quad \Longrightarrow \quad \Psi_\forall(f_1) \vdash \Psi_\forall(f_2)$$

In particular we have $f_1 \vdash f_2 \Longrightarrow \Psi(f_1) \vdash \Psi(f_2)$ for formulas f_1, f_2 without free variables.

Proof: The proof is canonically done by induction over the length of the proof for $f_1 \vdash f_2$, using Lem. 11.11. □

12 Summary of Beta

We have reached the following situation: We have two logical systems CG and FOL and two mappings Φ and Ψ between them. Each system has a derivability relation $\vdash$ and an entailment relation $\models$ (which are indexed in this chapter):

$$\begin{tikzcd}(\mathrm{FOL}, \vdash_{\mathrm{FOL}}, \models_{\mathrm{FOL}}) \arrow[r, shift left, "\Psi"] & (\mathrm{CG}, \vdash_{\mathrm{CG}}, \models_{\mathrm{CG}}) \arrow[l, shift left, "\Phi"]\end{tikzcd}$$

We are now prepared to work out how these mappings and relations are related to each other. This will be done in this chapter. We start with the analysis of the inner structure of both logical systems. It is easy to see that each of the relations $\vdash_{\mathrm{FOL}}$, $\models_{\mathrm{FOL}}$, $\vdash_{\mathrm{CG}}$, $\models_{\mathrm{CG}}$ is a quasiorder (i.e. a reflexive and transitive relation) on FOL resp. on CG. The calculus on FOL is sound and complete, i.e., $\vdash_{\mathrm{FOL}}$ and $\models_{\mathrm{FOL}}$, although they are defined in a very different way, are in fact the same relation. Of course, we expect that this will be the same for CG, too. This will be proven in Thm. 12.2.

Furthermore we have said that Φ and Ψ can be considered as translations between FOL and CG. As they are translations, they are not simple mappings between the *sets* of FOL and CG. Furthermore they have to preserve the *meaning* of the formulas resp. concept graphs. In other words: They have to respect the semantical entailment relations $\models_{\mathrm{FOL}}$ and $\models_{\mathrm{CG}}$. Furthermore we expect the following: If we translate a formula into a graph and if we then translate this graph back into a new formula, the original formula and the new formula should have the same meaning. The same should hold if we start with a concept graph. To summarize: If we say that Φ and Ψ are *translations* between the systems FOL and CG, they should be quasiorder-isomorphisms between $(\mathrm{FOL}, \models_{\mathrm{FOL}})$ and $(\mathrm{CG}, \models_{\mathrm{CG}})$ which are, up to equivalence, mutually inverse. This will be shown in Thm. 12.3.

F. Dau: The Logic System of Concept Graphs with Negation, LNAI 2892, pp. 157-162, 2003.

12.1 Summary and Main Results

If we resume all main theorems of the last chapters, we have:

Let $\mathfrak{G}$, $\mathfrak{G}_1$, $\mathfrak{G}_2$ be concept graphs over $\mathcal{A}$ and let f, f_1, f_2 be FOL-formulas over $\mathcal{A}$. Then the following conditions hold:

1)	$f \vdash \Phi(\Psi(f))$ and $\Phi(\Psi(f)) \vdash f$	Thm. 11.8
2)	$\mathfrak{G} \vdash \Psi(\Phi(\mathfrak{G}))$ and $\Psi(\Phi(\mathfrak{G})) \vdash \mathfrak{G}$	Thm. 11.9
3)	$f_1 \vdash f_2 \quad \Longrightarrow \quad \Psi(f_1) \vdash \Psi(f_2)$	Thm. 11.12
4)	$\mathfrak{G}_1 \vdash \mathfrak{G}_2 \quad \Longrightarrow \quad \mathfrak{G}_1 \models \mathfrak{G}_2$	Thm. 10.14
5)	$f_1 \vdash f_2 \quad \Longleftrightarrow \quad f_1 \models f_2$	Thm. 8.9
6)	$\mathfrak{G}_1 \models \mathfrak{G}_2 \quad \Longleftrightarrow \quad \Phi(\mathfrak{G}_1) \models \Phi(\mathfrak{G}_2)$	Thm. 9.10

The most extensive proof was the proof condition 3) (i.e., of Thm. 11.12) which states that Ψ respects the syntactical entailment relation $\vdash$ on FOL. But we have not shown the counterpart in the inverse direction, that is we have not proven the following condition:

Let $\mathfrak{G}_1$ and $\mathfrak{G}_2$ be two concept graphs over $\mathcal{A}$. Then we have

$$4') \quad \mathfrak{G}_1 \vdash \mathfrak{G}_2 \quad \Longrightarrow \quad \Phi(\mathfrak{G}_1) \vdash \Phi(\mathfrak{G}_2)$$

Instead of this condition we have proven condition 4) which states that the calculus $\vdash$ on CG is sound. But we can replace condition 4) by condition 4') and vice versa, as the following lemma shows.

Lemma 12.1 (Conditions 4) and 4') are Equivalent).
From 4), 5), 6) follows 4') and from 4'), 5), 6) follows 4).

Proof: First we assume that conditions 4), 5) and 6) are valid. We conclude

$$\mathfrak{G}_1 \vdash \mathfrak{G}_2 \overset{4)}{\Longrightarrow} \mathfrak{G}_1 \models \mathfrak{G}_2 \overset{6)}{\Longleftrightarrow} \Phi(\mathfrak{G}_1) \models \Phi(\mathfrak{G}_2) \overset{5)}{\Longleftrightarrow} \Phi(\mathfrak{G}_1) \vdash \Phi(\mathfrak{G}_2) \quad ,$$

hence 4') holds. If we assume on the other hand that 4'), 5) and 6) are valid, we conclude

$$\mathfrak{G}_1 \vdash \mathfrak{G}_2 \overset{4')}{\Longrightarrow} \Phi(\mathfrak{G}_1) \vdash \Phi(\mathfrak{G}_2) \overset{5)}{\Longleftrightarrow} \Phi(\mathfrak{G}_1) \models \Phi(\mathfrak{G}_2) \overset{6)}{\Longleftrightarrow} \mathfrak{G}_1 \models \mathfrak{G}_2 \quad ,$$

hence 4) holds. □

Condition 5) (i.e., Thm. 8.9) states that $\models_{\mathrm{FOL}}$ is sound and complete, that is $\vdash_{\mathrm{FOL}}$ and $\models_{\mathrm{FOL}}$ are in fact the same relation. Now we can prove that the same holds for $\vdash_{\mathrm{CG}}$ and $\models_{\mathrm{CG}}$.

Theorem 12.2 (Sound- and Completeness of the Beta-Calculus). *Let $\mathfrak{G}_1$ and $\mathfrak{G}_2$ be concept graphs over $\mathcal{A}$. Then we have*

$$\mathfrak{G}_1 \vdash \mathfrak{G}_2 \quad \Longleftrightarrow \quad \mathfrak{G}_1 \models \mathfrak{G}_2$$

Proof: We only have to show direction '$\Longleftarrow$'. We have

$$\begin{aligned}\mathfrak{G}_1 \models \mathfrak{G}_2 &\overset{6)}{\Longrightarrow} \Phi(\mathfrak{G}_1) \models \Phi(\mathfrak{G}_2)\\ &\overset{5)}{\Longrightarrow} \Phi(\mathfrak{G}_1) \vdash \Phi(\mathfrak{G}_2)\\ &\overset{3)}{\Longrightarrow} \Psi(\Phi(\mathfrak{G}_1)) \vdash \Psi(\Phi(\mathfrak{G}_2))\\ &\overset{2)}{\Longrightarrow} \mathfrak{G}_1 \vdash \mathfrak{G}_2 \qquad \Box\end{aligned}$$

Conditions 1)–3) and 4') state the Φ : FOL → CG and Ψ : FOL → CG are, up to equivalence, mutually inverse isomorphisms between the quasiordered sets (FOL, $\vdash$) and (CG, $\vdash$). Furthermore we have $\vdash_{\mathrm{FOL}} = \models_{\mathrm{FOL}}$ and $\vdash_{\mathrm{CG}}$ $= \models_{\mathrm{CG}}$, hence we get immediately the following theorem:

Corollary 12.3 (Main Quasiorder Theorem for FOL and CG). *Φ : FOL → CG and Ψ : FOL → CG are, up to equivalence, mutually inverse isomorphisms between the quasiordered sets (FOL, $\vdash$) and (CG, $\vdash$) and between the quasiordered sets (FOL, $\models$) and (CG, $\models$).*

Without proof.

The last two results can be regarded the main results of this treatise. Together with the well-known Thm. 8.9, we get the full syntactical and semantical equivalence between FOL and CG.

12.2 Independency Results

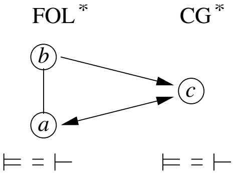
In the last section we have argued that the conditions 1)–6) yield the equivalence between FOL and CG. In this section we want to explain why it is not only sufficient, but also necessary to prove conditions 1)–6). To put it more formally, we will show the following: None of the conditions 1)–6) can be derived from the remaining five conditions by only using simple arguments, that means arguments which only use the fact that all relations are quasiorders.

Theorem 12.4 (Independency Results for Conditions 1)–6)). *Consider CG and FOL as two sets, each of them equipped with two quasiorders $\models$ and $\vdash$. Further let Φ : CG → FOL and Ψ : FOL → CG be two mappings. We understand conditions 1)–6) as conditions for these quasiorders and mappings (with $\mathfrak{G}, \mathfrak{G}_1, \mathfrak{G}_2 \in$ CG and $f, f_1, f_2 \in$ FOL). Then none of the conditions 1)–6) can be derived from the remaining five.*

Proof: For each of the conditions 1)–6), we have to give an example such that the selected condition does not hold in this example, but the remaining conditions do.

1) Let $\mathrm{FOL}^* := \{a, b\}$ and $\mathrm{CG}^* := \{c\}$. We set

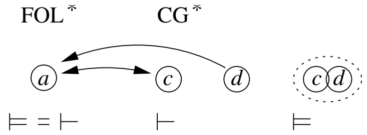
- $\vdash_{\mathrm{FOL}^*} := \models_{\mathrm{FOL}^*} := \{(a,a),(b,b),(a,b)\}$
- $\vdash_{\mathrm{CG}^*} := \models_{\mathrm{CG}^*} := \{(c,c)\}$
- $\Psi(a) := \Psi(b) := c$ and $\Phi(c) := a$



We can visualise this situation as above. The mappings $\Psi : \mathrm{FOL}^* \to \mathrm{CG}^*$ and $\Phi : \mathrm{CG}^* \to \mathrm{FOL}^*$ are indicated by the arrows, and the diagrams for the quasiorders are drawn as usual. It is easy to see that conditions 2)–6) hold in this example. But we have $b \not\vdash a = \Phi(\Psi(b))$, hence condition 1) is not valid. Therefore 1) cannot be derived from 2)–6).

2) This is the only example where we need on one side a quasiorder (see Lem. 12.5). Let $\mathrm{FOL}^* := \{a\}$ and $\mathrm{CG}^* := \{b, c\}$. We set

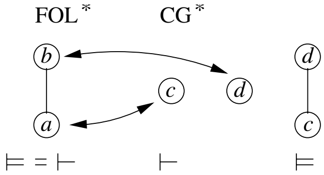
- $\vdash_{\mathrm{FOL}^*} := \models_{\mathrm{FOL}^*} := \{(a,a)\}$
- $\vdash_{\mathrm{CG}^*} := \{(c,c),(d,d)\}$
- $\models_{\mathrm{CG}^*} := \{(c,c),(d,d),(c,d),(d,c)\}$
- $\Psi(a) := c$
- $\Phi(c) := a$, $\Phi(d) := a$



We can visualise this situation as above. As $\vdash_{\mathrm{CG}^*}$ and $\models_{\mathrm{CG}^*}$ are different, we have to give a diagram for each relation. In this example, we have $d \not\vdash c = \Psi(\Phi(d))$, hence condition 2) is not valid. But it is easy to see that all remaining conditions hold.

3) Let $\mathrm{FOL}^* := \{a, b\}$ and $\mathrm{CG}^* := \{c, d\}$. We set

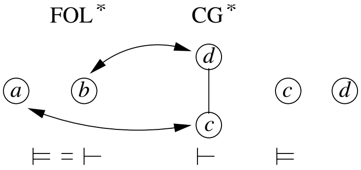
- $\vdash_{\mathrm{FOL}^*} := \models_{\mathrm{FOL}^*} := \{(a,a),(b,b),(a,b)\}$
- $\vdash_{\mathrm{CG}^*} := \{(c,c),(d,d)\}$
- $\models_{\mathrm{CG}^*} := \{(c,c),(d,d),(c,d)\}$
- $\Psi(a) := c$, $\Psi(b) := d$
- $\Phi(c) := a$, $\Phi(d) := b$



In this example we have $a \vdash b$, but $\Psi(a) = c \not\vdash d = \Psi(b)$. So condition 3) is not valid, but all remaining conditions hold in this example.

4) Let FOL* := $\{a, b\}$ and CG* := $\{c, d\}$. We set

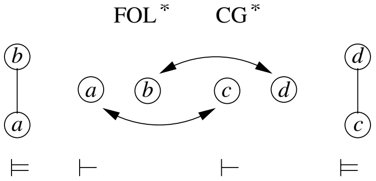
- $\vdash_{\mathrm{FOL}^*} := \models_{\mathrm{FOL}^*} := \{(a,a),(b,b)\}$
- $\vdash_{\mathrm{CG}^*} := \{(c,c),(d,d),(c,d)\}$
- $\models_{\mathrm{CG}^*} := \{(c,c),(d,d)\}$
- $\Psi(a) := c$, $\Psi(b) := d$
- $\Phi(c) := a$, $\Phi(d) := b$



Here we have $c \vdash d$, but $c \not\models d$. So condition 4) is not valid, but all remaining conditions hold in this example.

5) Let FOL* := $\{a, b\}$ and CG* := $\{c, d\}$. We set

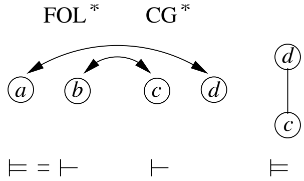
- $\models_{\mathrm{FOL}^*} := \{(a,a),(b,b),(a,b)\}$
- $\vdash_{\mathrm{FOL}^*} := \{(a,a),(b,b)\}$
- $\models_{\mathrm{CG}^*} := \{(c,c),(d,d),(c,d)\}$
- $\vdash_{\mathrm{CG}^*} := \{(c,c),(d,d)\}$
- $\Psi(a) := c$, $\Psi(b) := d$
- $\Phi(c) := a$, $\Phi(d) := b$



In this example we have $a \models b$ and $a \not\vdash b$. So the direction '$\Longleftarrow$' of condition 5) is not fulfilled. But all remaining conditions hold in this example.

6) Let FOL* := $\{a, b\}$ and CG* := $\{c, d\}$. We set

- $\vdash_{\mathrm{FOL}^*} := \models_{\mathrm{FOL}^*} := \{(a,a),(b,b)\}$
- $\models_{\mathrm{CG}^*} := \{(c,c),(d,d),(c,d)\}$
- $\vdash_{\mathrm{CG}^*} := \{(c,c),(d,d)\}$
- $\Psi(a) := d$, $\Psi(b) := c$
- $\Phi(c) := b$, $\Phi(d) := a$



In this example we have $c \models d$ and $\Phi(c) = b \not\models a = \Phi(d)$. So the direction '$\Longrightarrow$' of condition 6) is not fulfilled. But all remaining conditions hold in this example. □

Note that condition 2) was the sole condition which required a quasiorder in the above counterexamples. More specific: In the examples above, all relations $\vdash_{\mathrm{FOL}^*}$, $\models_{\mathrm{FOL}^*}$, $\models_{\mathrm{CG}^*}$, $\vdash_{\mathrm{CG}^*}$ are orders, except for $\models_{\mathrm{CG}^*}$ in the counterexample for condition 2). This is no accident, as the following lemma shows.

Lemma 12.5 (Conditions 2) can be Derived for Orders).

If $\models_{CG^*}$ is an order, than we can derive condition 2) from conditions 1), 5) and 6).

Proof: Let $\mathfrak{G}$ be a concept graph. For $f := \Phi(\mathfrak{G})$ we conclude $\Phi(\Psi(f)) \vdash f$ and $f \vdash \Phi(\Psi(f))$ from condition 1). Hence condition 5) yields $\Phi(\Psi(f)) \models f$ and $f \models \Phi(\Psi(f))$, i.e., $\Phi(\Psi(\Phi(\mathfrak{G}))) \models \Phi(\mathfrak{G})$ and $\Phi(\mathfrak{G}) \models \Phi(\Psi(\Phi(\mathfrak{G})))$. Now we apply condition 6) to both entailment relations and get $\Psi(\Phi(\mathfrak{G})) \models \mathfrak{G}$ and $\mathfrak{G} \models \Psi(\Phi(\mathfrak{G}))$. The antisymmetry of $\models_{\mathrm{CG}^*}$ yields $\Psi(\Phi(\mathfrak{G})) = \mathfrak{G}$, thus condition 6) is fulfilled. □

13 Concept Graphs without Cuts

Reasoning on concept graphs is carried out by the transformation rules of the calculus. If we consider only concept graphs *without* cuts (which correspond to the existential-conjunctive fragment of FOL), further possibilities – which are based on ideas presented in Prediger's PhD-thesis (see [44]) – for doing reasoning are possible. These possibilities will be described in this chapter.

In [44], Prediger developped a mathematical theory for concept graphs without cuts. As in Prediger's graphs negation cannot be expressed, these graphs correspond to the existential-conjunctive fragment of conceptual graphs or of FOL. Prediger performed reasoning on these graphs in two different ways: First of all, she introduced a sound and complete calculus (which consists of fairly simple transformation rules). Secondly, she assigned to each graph a corresponding *standard model* (and, vice versa, to each model a corresponding *standard graph*). For graphs without generic markers, Prediger's standard models encode exactly the same information as the respective graphs. Thus, for these graphs, the reasoning on graphs can be carried over (in a sense which will be made clear in the next sections) to the models.

In this chapter, we bring together the ideas of Prediger and Dau for concept graphs without cuts. First of all, we consider the restricted version of the calculus of Chap. 10, where we have removed all rules where cuts are involved. Secondly, we extend Prediger's notions of standard graphs and standard models (the differences to Prediger's approach will be discussed in the next sections) such that even for graphs *with* generic markers, their standard models will encode exacty the same exactly the same information as the respective graphs. On the models, we introduce a semantical entailment relation $\models$ as well as transformation rules. The latter can be seen as a kind of calculus which yields a relation $\vdash$ between models. It will turn out that the relations $\models, \vdash$ for graphs and for models and the notions of standard models and standard graphs fit perfectly together. That is, both calculi are adequate and reasoning can be carried over from graphs to models and vice versa.

F. Dau: The Logic System of Concept Graphs with Negation, LNAI 2892, pp. 163-185, 2003.

13.1 Concept Graphs without Cuts

In this chapter, we only consider concept graphs without cuts, i.e. graphs $\mathfrak{G} := (V, E, \nu, \top, Cut, area, \kappa, \rho)$ with $Cut = \emptyset = area$. For this reason, we will write $\mathfrak{G} := (V, E, \nu, \kappa, \rho)$ instead (as we do not consider contexts, we have removed the sheet of assertion $\top$ from the notation of simple concept graphs as well). Although these graphs are similar to the simple concept graphs as they are defined by Prediger in [44], there are some important syntactical differences between these two kinds of graphs.

First of all, in [44] Prediger assigned *sets* of objects instead of *single* objects to vertices (i. e. in [44] we have $\rho : V \to \mathfrak{P}(\mathcal{G}) \dot\cup \{*\}$ instead of $\rho : V \to \mathcal{G} \dot\cup \{*\}$). For concept graphs with cuts, it is not immediately clear what the meaning of a vertex is which is enclosed by a cut and which contains more than one object. For this reason, ρ assigns single objects to vertices. The expressiveness of the graphs is not changed by this syntactical restriction.

Identity is in [44] expressed by an equivalence relation θ only on the set of generic vertices. In this treatise, identity is expressed by identity links on the set of generic *and non-generic* vertices. Thus the concept graphs of this chapter have a slightly higher expressiveness than the concept graphs of [44]. This has to be taken into account in the definition of standard models and standard graphs, as well as in the calculus. To provide an example: Consider an alphabet with $\mathcal{C} := \{\top, A, B\}$ and $\mathcal{G} := \{a, b\}$, where A, B are incomparable concept names. In our approach, $\boxed{A\colon a}-(=)-\boxed{B\colon b}$ is a well-defined graph[1] which expresses a proposition which cannot represented in Prediger's approach. This graph entails $\boxed{A\colon b}-(=)-\boxed{B\colon a}$. Obviously, this entailment is based on the unique meaning of identity (for this reason, we have rules in our calculus which capture the role of the identity relation and which allow to derive the second graph from the first one). A derivation like this cannot be performed with projections[2] ([7]) or with the calculus presented in [38].

The calculus for concept graphs with cuts consists of the following rules: erasure, insertion, iteration, deiteration, double cuts, generalization, specialization, isomorphism, exchanging references, merging two vertices, splitting a vertex, $\top$-erasure, $\top$-insertion, identity-erasure and identity-insertion.

Only the double-cut-rule allows to derive a concept graph with cuts from a concept graph without cuts. The rules insertion and specialization can only be applied if we have a graph with cuts. As we consider concept graphs

[1] In contrast to other authors, like Mugnier [38], we allow that different object names may refer to the same object, i.e. we do not adopt the *unique name assumption.* The unique name assumption is needed when a graph shall be transformed into its normal-form. This graph cannot be transformed into a normalform and is not a well-defined graph in the approach of Mugnier.

[2] Moreover, as projections rely on normal-forms for graphs, they have further restrictions. See [38].

without cuts, we remove these three rules from the calculus and interpret the remaining rules as rules for the system of concept graphs without cuts. So we have the following definition (for examples, an explanation and a precise mathematical definition for the rules, we refer to [15]):

Definition 13.1 (Calculus for Concept Graphs without Cuts).

The calculus for concept graphs $\mathfrak{G} := (V, E, \nu, \kappa, \rho)$ *over the alphabet* $\mathcal{A} := (\mathcal{G}, \mathcal{C}, \mathcal{R})$ *consists of the following rules:*

Erasure, iteration, deiteration, generalization, isomorphism, exchanging references, merging two vertices, splitting a vertex, $\top$*-erasure,* $\top$*-insertion, identity-erasure and identity-insertion. If* $\mathfrak{G}_a$*,* $\mathfrak{G}_b$ *are concept graphs such that* $\mathfrak{G}_b$ *can be derived from* $\mathfrak{G}_a$*, we will write* $\mathfrak{G}_a \vdash_{pos} \mathfrak{G}_b$*.*

13.2 Standard Models and Semantical Entailment

In this section, we will assign to each graph its *standard model* which encodes exactly the same information as the graph. This is based on [75] and has already done by Prediger in Def. 4.2.5. of [44]. Remember that identity is in [44] expressed by an equivalence relation θ only on the set of generic vertices, so Prediger used the following approach: The set of objects of the standard model consists of all object names $G \in \mathcal{G}$ and of all equivalence classes of θ. But we can express identity between arbitrary, i. e. generic or non-generic, vertices, thus we have to extend this idea. We start by defining an equivalence relation $\theta_{\mathcal{G}}$ on $V \dot{\cup} \mathcal{G}$, which is an appropriate generalization of Prediger's θ.

Definition 13.2 (Equivalence-Relation $\boldsymbol{\theta_{\mathfrak{G}}}$ on $\boldsymbol{V \dot{\cup} \mathcal{G}}$).

Let $\mathfrak{G} := (V, E, \nu, \kappa, \rho)$ *be a concept graph over* $\mathcal{A}$*. We assume that* V *and* $\mathcal{G}$ *are disjoint. Let* $\theta_{\mathfrak{G}}$ *be the smallest equivalence relation on* $V \dot{\cup} \mathcal{G}$ *such that*

1. *if* $\rho(v) = G \in \mathcal{G}$*, then* $v\theta_{\mathfrak{G}}G$*, and*
2. *if* $e \in E$ *with* $\nu(e) = (v_1, v_2)$ *and* $\kappa(e) \leq \doteq$*, then* $v_1\theta_{\mathfrak{G}}v_2$*.*

It is easy to see that two vertices which are equivalent must refer in each model to the same object, i. e. we have the following lemma:

Lemma 13.3 ($\boldsymbol{\theta_{\mathfrak{G}}}$-Related Vertices Refer to Same Objects).

Let $\mathfrak{G} := (V, E, \nu, \kappa, \rho)$ *be a concept graph over* $\mathcal{A}$*. If* $v_1, v_2 \in V$ *with* $v_1\theta_{\mathfrak{G}}v_2$*, then* $ref(v_1) = ref(v_2)$ *for each contextual structure* $(\vec{\mathbb{K}}, \lambda)$ *over* $\mathcal{A}$ *and each valuation* $ref : V \to G_0$ *with* $(\vec{\mathbb{K}}, \lambda) \models \mathfrak{G}[ref]$*.*

The opposite direction of the lemma holds as well, i. e. we could characterize $\theta_{\mathfrak{G}}$ by the condition in the lemma. This is not immediately clear, but it could easily be shown with the results of this chapter.

Now we can assign to each concept graph an appropriate standard model which encodes exactly the same information as the graph.[3]

Definition 13.4 (Standard Model).

Let $\mathfrak{G} := (V, E, \nu, \kappa, \rho)$ be a concept graph over $\mathcal{A}$. We define the standard model of $\mathfrak{G}$ *as follows:*

For $\mathfrak{G} \neq \emptyset$ or $\mathcal{G} \neq \emptyset$,[4] we first define a power context family $\vec{\mathbb{K}}^{\mathfrak{G}}$ by

- *$G_0^{\mathfrak{G}} := \{[k]\theta_{\mathfrak{G}} \mid k \in V \dot{\cup} \mathcal{G}\}$, and $G_i^{\mathfrak{G}} := (G_0^{\mathfrak{G}})^i$,*
- *$M_0^{\mathfrak{G}} := \mathcal{C}$, and $M_i^{\mathfrak{G}} := \mathcal{R}_i$ for $1 \leq i \leq n$.*
- *For $C \in \mathcal{C}$ and $g \in G_0$, we set*

$$gI_0^{\mathfrak{G}}C \quad :\Longleftrightarrow \quad C = \top \text{ or } \exists v \in V \,.\, g = [v]\theta_{\mathfrak{G}} \wedge \kappa(v) \leq C\,.$$

- *For $R_i \in \mathcal{R}_i$ and $g_1, \ldots, g_i \in G_0$, we set $(g_1, \ldots, g_i)I_i^{\mathfrak{G}}R_i :\Longleftrightarrow$*
  - *$\exists e = (v_1, \ldots, v_i) \in E \,.\, g_1 = [v_1]\theta_{\mathfrak{G}} \wedge \ldots \wedge g_i = [v_i]\theta_{\mathfrak{G}} \wedge \kappa(e) \leq R_i$, or*
  - *$i = 2$ and $\doteq \leq R_i \wedge g_1 = g2$.*

The mappings $\lambda^{\mathfrak{G}}$ are defined canonically: For each $G \in \mathcal{G}$, $C \in \mathcal{C}$ and $R \in \mathcal{R}$ we set

$$\lambda_{\mathcal{G}}^{\mathfrak{G}}(G) := [G]\theta_{\mathfrak{G}} \quad , \quad \lambda_{\mathcal{C}}^{\mathfrak{G}}(C) := \mu(C) \quad , \text{ and } \quad \lambda_{\mathcal{R}}^{\mathfrak{G}}(R) := \mu(R).$$

If $\mathfrak{G} = \emptyset$ and $\mathcal{G} = \emptyset$, let g be an arbitrary element. We define $\vec{\mathbb{K}}^{\mathfrak{G}}$ as follows: $\mathbb{K}_0 := (\{g\}, \{\top\}, \{(g, \top)\})$, $\mathbb{K}_2 := (\{(g, g)\}, \{\doteq\}, \{((g, g), \doteq)\})$, and we set $\mathbb{K}_i := (\emptyset, \emptyset, \emptyset)$ for $i \neq 0, 2$. The mappings of $\lambda^{\mathfrak{G}}$ are defined canonically, i. e. $\lambda_{\mathcal{G}}^{\mathfrak{G}} := \emptyset$, $\lambda_{\mathcal{C}}^{\mathfrak{G}}(\top) := \mu(\top)$, and $\lambda_{\mathcal{R}}^{\mathfrak{G}}(\doteq) := \mu(\doteq)$. All remaining concept- or relation-names are mapped to the $\bot$-concept $(\emptyset'', \emptyset')$ of the respective formal context.

[3] This is possible because we consider only the existential-conjunctive fragment of concept graphs (in particular we do not consider negations or disjunctions of propositions), so we have to encode only the information a) whether objects have specific properties or whether objects are in a specific relation, b) whether objects *exist* with specific properties, and c) the conjunction of informations like these. These are the kinds of information which can be expressed in graphs (e.g. existential graphs or concept graphs) in an iconical way, i. e. we encode in standard models exactly the iconical features of concept graphs. For a deep discussion of this topic, we refer to [56].

[4] As usual in logic, we only consider non-empty structures. For this reason, we have to treat the case $\mathfrak{G} = \emptyset = \mathcal{G}$ separately.

The contextual structure $(\vec{\mathbb{K}}^{\mathfrak{G}}, \lambda^{\mathfrak{G}})$ *is called* standard model of $\mathfrak{G}$ *and is denoted by* $\mathcal{M}^{\mathfrak{G}}$.[5]

It is not immediately clear that the definition above yields indeed a contextual structure. Of course $\vec{\mathbb{K}}^{\mathfrak{G}}$ is a power context family. It remains to check that $\lambda := \lambda^{\mathfrak{G}}_{\mathcal{G}} \dot{\cup} \lambda^{\mathfrak{G}}_{\mathcal{C}} \dot{\cup} \lambda^{\mathfrak{G}}_{\mathcal{R}}$ fulfills the conditions of Def. 4.3. This is done now.

1. We have to show that $\lambda^{\mathfrak{G}}_{\mathcal{C}}$ and $\lambda^{\mathfrak{G}}_{\mathcal{R}}$ are order-preserving. We only consider the mapping $\lambda^{\mathfrak{G}}_{\mathcal{R}}$ (the case $\lambda^{\mathfrak{G}}_{\mathcal{C}}$ is done analogously). So let $R_1, R_2 \in \mathcal{R}_i$ and $(g_1, \ldots, g_i) \in \mathrm{Ext}(\lambda^{\mathfrak{G}}_{\mathcal{R}}(R_1))$. If there is an $e = (v_1, \ldots, v_i) \in E$ which satisfies $g_1 = [v_1]\theta_{\mathfrak{G}} \wedge \ldots \wedge g_i = [v_i]\theta_{\mathfrak{G}}$ and $\kappa(e) \leq R_1$, we have $\kappa(e) \leq R_2$ as well, so, by Def. 4.3, we conclude $(g_1, \ldots, g_i) \in \mathrm{Ext}(\lambda^{\mathfrak{G}}_{\mathcal{R}}(R_2))$. If we have $i = 2$, $\doteq \leq R_1$ and $g_1 = g2$, then we have $\doteq \leq R_2$ as well, so, again by Def. 4.3, we conclude $(g_1, g_2) \in \mathrm{Ext}(\lambda^{\mathfrak{G}}_{\mathcal{R}}(R_2))$.
2. It is easy to see that $\lambda^{\mathfrak{G}}_{\mathcal{C}}(\top) = \top$ holds.
3. It remains to show: $(g_1, g_2) \in \mathrm{Ext}(\lambda_{\mathcal{R}}\mathfrak{G}(\doteq)) \Leftrightarrow g_1 = g_2$ for all $g_1, g_2 \in G_0$. The direction '$\Longleftarrow$' is easy to see: For $g \in G_0$, the second condition for R_i of Def. 4.3, applied to $i := 2$ and $R_i :=\doteq$ yields $(g, g)I^{\mathfrak{G}}_2 \doteq$. In order to show direction '$\Longrightarrow$', we assume that we have $g_1, g_2 \in G_0$ and an edge $e = (v_1, v2) \in E$ with $g_1 = [v_1]\theta_{\mathfrak{G}}$, $g_2 = [v_2]\theta_{\mathfrak{G}}$ and $\kappa(e) \leq\doteq$. Def. 13.2 yields $v_1\theta_{\mathfrak{G}}v_2$, from which we conclude $g_1 = [v_1]\theta_{\mathfrak{G}} = [v_2]\theta_{\mathfrak{G}} = g_2$.

As all conditions of Def. 4.3 are fulfilled, we see that $(\vec{\mathbb{K}}^{\mathfrak{G}}, \lambda^{\mathfrak{G}})$ is in fact a conctextual structure. Furthermore, it is easy to see that each graph holds in its standard model, i. e. we have:

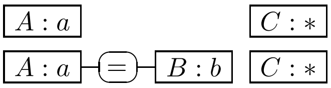
Lemma 13.5 (Each Graph Holds in its Standard Model).

If $\mathfrak{G} := (V, E, \nu, \kappa, \rho)$ *is a graph, then* $(\vec{\mathbb{K}}^{\mathfrak{G}}, \lambda^{\mathfrak{G}})$ *is a contextual structure over* $\mathcal{A}$. *For* $ref^{\mathfrak{G}} := \{(v, [v]\theta_{\mathfrak{G}}) \mid v \in V\}$, *we have* $(\vec{\mathbb{K}}^{\mathfrak{G}}, \lambda^{\mathfrak{G}}) \models \mathfrak{G}[ref^{\mathfrak{G}}]$.

Without proof.

In the following, we provide some examples for simple concept graphs (over the alphabet $(\{a, b, c\}, \{A, B, C, \top\}, \{\doteq\})$ with incomparable concept names A, B, C and their standard models.

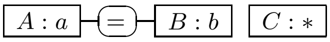
1. $\mathfrak{G}_1 :=$



$\mathcal{M}_{\mathfrak{G}_1} :=$

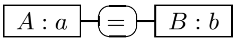
	A	B	C	$\top$
$\{a, b, v_1, v_2, v_4\}$	×	×		×
$\{v_3\}$			×	×
$\{v_5\}$			×	×

[5] We write $\mathcal{M}^{\mathfrak{G}}$ instead of $\mathcal{M}_{\mathfrak{G}}$, because the different contexts of $\vec{\mathbb{K}}^{\mathfrak{G}}$ and the mappings of $\lambda^{\mathfrak{G}}$ already have indices at the bottom.

2. $\mathfrak{G}_2 :=$ 

$\mathcal{M}_{\mathfrak{G}_2} :=$

	A	B	C	$\top$
$\{a, b, v_1, v_2\}$	×	×		×
$\{v_3\}$			×	×

3. $\mathfrak{G}_3 :=$ 

$\mathcal{M}_{\mathfrak{G}_3} :=$

	A	B	C	$\top$
$\{a, b, v_1, v_2\}$	×	×		×

4. $\mathfrak{G}_4 :=$ $\boxed{A : a}$ $\boxed{B : b}$ $\boxed{C : *}$

$\mathcal{M}_{\mathfrak{G}_2} :=$

	A	B	C	$\top$
$\{a, v_1\}$	×			×
$\{b, v_2\}$		×		×
$\{v_3\}$			×	×

5. $\mathfrak{G}_5 :=$ $\boxed{A : a}$ $\boxed{B : b}$

$\mathcal{M}_{\mathfrak{G}_5} :=$

	A	B	C	$\top$
$\{a, v_1\}$	×			×
$\{b, v_2\}$		×		×

6. $\mathfrak{G}_6 :=$ $\boxed{A : a}$

$\mathcal{M}_{\mathfrak{G}_6} :=$

	A	B	C	$\top$
$\{a, v_1\}$	×			×
$\{b\}$				×

7. $\mathfrak{G}_7 :=$

$\mathcal{M}_{\mathfrak{G}_7} :=$

	A	B	C	$\top$
$\{a\}$				×
$\{b\}$				×

The well-known relation $\models$ on graphs can be understood as follows: $\mathfrak{G}_1 \models \mathfrak{G}_2$ holds iff $\mathfrak{G}_1$ contains the same or more information than $\mathfrak{G}_2$. This idea can be transferred to models as well. This yields the following definition:[6]

Definition 13.6 (Sem. Entailment Between Context. Structures).

Let $\mathcal{M}^a := (\vec{\mathbb{K}}^a, \lambda^a)$ and $\mathcal{M}^b := (\vec{\mathbb{K}}^b, \lambda^b)$ be two contextual structures structures over $\mathcal{A}$ (with $\vec{\mathbb{K}}^x = (\mathbb{K}_0^x, \ldots, \mathbb{K}_n^x)$ and $\mathbb{K}_i^x = (G_i^x, M_i^x, I_i^x)$, $i = 1, \ldots, n$ and $x = a, b$). We set $\mathcal{M}^a \models \mathcal{M}^b$ if and only if there is a mapping $f : G_0^b \to G_0^a$ which fulfills the following conditions:

1. *For all $G \in \mathcal{G}$: $f(\lambda_\mathcal{G}^b(G)) = \lambda_\mathcal{G}^a(G)$ (i. e. f respects $\lambda_\mathcal{G}$.)*
2. *For all $g \in G_0^b$ and $C \in \mathcal{C}$: $g \in Ext(\lambda_\mathcal{C}^b(C)) \Longrightarrow f(g) \in Ext(\lambda_\mathcal{C}^a(C))$ (i. e. f respects $\lambda_\mathcal{C}$).*
3. *For all $\vec{g} \in G_i^b$ and $R \in \mathcal{R}_i$: $\vec{g} \in Ext(\lambda_\mathcal{R}^b(R)) \Longrightarrow f(\vec{g}) \in Ext(\lambda_\mathcal{R}^a(R))$ where $f(\vec{g}) = f(g_1, \ldots, g_i) := (f(g_1), \ldots, f(g_i))$ (i. e. f respects $\lambda_\mathcal{R}$).*

We will sometimes write $\mathcal{M}^a \models_f \mathcal{M}^b$ to denote the mapping f, too.

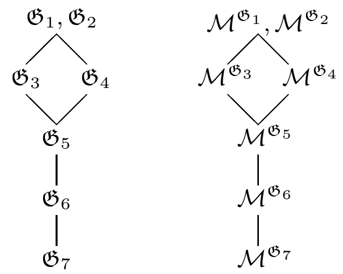
[6] See also [80], where Wille defines in a different way the *informational content* of a model.

We want to make the following remarks on standard models:

1. The direction of f is not surprising: It corresponds to the direction of projections between conceptual graphs (see [7]).[7] In fact, f can be roughly understood as projection between models (instead of between graphs). But note that projection between graphs is complete only under certain restrictions (e.g., on the normal form of one graph. But, as already argued, not every graph can be transformed into a normal-form), but it will turn out that projection between models is complete without further restrictions. Thus, to evaluate whether a graph $\mathfrak{G}_a$ entails a graph $\mathfrak{G}_b$, a sound and complete approach is the following: First contruct the standard models $\mathcal{M}_{\mathfrak{G}_a}$ and $\mathcal{M}_{\mathfrak{G}_b}$, and then find out whether there is a 'projection' f from $\mathcal{M}_{\mathfrak{G}_b}$ to $\mathcal{M}_{\mathfrak{G}_a}$.
2. Please note that the two models below are semantically equivalent, although they have a different number of objects. This is based on the fact that we cannot count in existential-conjunctive languages (without negation, we cannot express that we have *different* objects with the same properties).

$$\mathcal{M}_1 := \begin{array}{|c||c|c|}\hline \mathbb{K}_0^1 & P & \top \\ \hline\hline g & \times & \times \\ \hline\end{array} \qquad\qquad \mathcal{M}_2 := \begin{array}{|c||c|c|}\hline \mathbb{K}_0^2 & P & \top \\ \hline\hline g & \times & \times \\ \hline h & \times & \times \\ \hline\end{array}$$

3. Concerning the relation $\models$ on the set of graphs resp. on the set of contextual structures, the graphs and their standard models above are ordered as follows:


$\mathfrak{G}_1, \mathfrak{G}_2$
$\mathfrak{G}_3$
$\mathfrak{G}_4$
$\mathfrak{G}_5$
$\mathfrak{G}_6$
$\mathfrak{G}_7$
$\mathcal{M}^{\mathfrak{G}_1}, \mathcal{M}^{\mathfrak{G}_2}$
$\mathcal{M}^{\mathfrak{G}_3}$
$\mathcal{M}^{\mathfrak{G}_4}$
$\mathcal{M}^{\mathfrak{G}_5}$
$\mathcal{M}^{\mathfrak{G}_6}$
$\mathcal{M}^{\mathfrak{G}_7}$

4. In [44], standard models are compared as well, but Prediger compares only the restrictions of the incidence-relations to objects which are generated by non-generic nodes. E.g. in Prediger's approach, the standard models of the graphs $\boxed{A : *}$ and $\boxed{B : *}$ are comparable, although they encode incomparable information. Thus Prediger's approach is strictly weaker than semantical entailment between models, as it is elaborated in this chapter.

[7] Another comparison can be drawn to algebra: A Standard Model of a graph can be compared with a free structure over a set of equations in an algebra, and every algebra which fulfills the equations can be mapped homomorphic into the free algebra.

The first step to show that reasoning can be carried over from graphs to their standard models (and vice vice) will be the next theorem which says that it makes no difference whethera graph or its standard model is evaluated in another model. This theorem is the main (semantical) link between a graph and its standard model. As Prediger has no concept of semantical entailment between models, there is no corresponding theorem in [44].

Theorem 13.7 (Main Thm. for Graphs and Standard Models).

Let $\mathfrak{G} := (V, E, \nu, \kappa, \rho)$ be a graph, $\mathcal{M}^{\mathfrak{G}} := (\vec{\mathbb{K}}^{\mathfrak{G}}, \lambda^{\mathfrak{G}})$ be its standard model and $\mathcal{M} := (\vec{\mathbb{K}}, \lambda)$ be an arbitrary contextual structure. Then we have

$$\mathcal{M} \models \mathcal{M}^{\mathfrak{G}} \quad \Longleftrightarrow \quad \mathcal{M} \models \mathfrak{G} \; .$$

Proof: We start with the direction '$\Longrightarrow$'. We have $\mathfrak{G} \xrightarrow{ref^{\mathfrak{G}}} \mathcal{M}^{\mathfrak{G}} \xrightarrow{f} \mathcal{M}$. Let $ref := f \circ ref^{\mathfrak{G}}$. We want to show that ref is a valuation with $\mathcal{M} \models \mathfrak{G}[ref]$. First we have to check that the mapping ref is indeed a valuation. So let $v \in V^{\mathcal{G}}$ with $\rho(v) = G \in \mathcal{G}$. We have

$$ref(v) = f(ref^{\mathfrak{G}}(v)) \overset{\text{Def for val.}}{=} f(\lambda_{\mathcal{G}}^{\mathfrak{G}}(\rho(v)) \overset{\text{Def 13.6}}{=} \lambda_{\mathcal{G}}(\rho(v)) \; ,$$

thus ref is a valuation.

To show $\mathcal{M} \models \mathfrak{G}[ref]$, we have to check that ref satisfies the vertex- and edge-conditions of [15], Def. 10.3.

To see that all vertex-conditions are fulfilled, let $v \in V$ be a vertex. Let $\kappa(v) = C \in \mathcal{C}$. From $\mathcal{M}^{\mathfrak{G}} \models \mathfrak{G}[ref^{\mathfrak{G}}]$ we conclude $ref^{\mathfrak{G}}(v) \in \mathrm{Ext}(\lambda_{\mathcal{G}}^{\mathfrak{G}}(C))$. Now condition 2) for f yields $ref(v) = f(ref^{\mathfrak{G}}(v)) \in \mathrm{Ext}(\lambda_{\mathcal{G}}(C))$, hence ref satisfies the vertex condition for v.

The edge-conditions are shown analogously, which completes the proof for the direction '$\Longrightarrow$'.

Now we show the direction '$\Longleftarrow$' of this lemma.

We have a valuation $ref : V \to G_0$ (where G_0 is the set of objects in $\mathcal{M} = (\vec{\mathbb{K}}, \lambda)$) with $\mathcal{M} \models \mathfrak{G}(ref)$. Furthermore we have the canonical valuation $ref^{\mathfrak{G}} : V \to G_0^{\mathfrak{G}}$ (where $G_0^{\mathfrak{G}}$ is the set of objects in $\mathcal{M}^{\mathfrak{G}}$) with $\mathcal{M}^{\mathfrak{G}} \models \mathfrak{G}(ref^{\mathfrak{G}})$. Let $(ref^{\mathfrak{G}})^{-1}$ be an inverse mapping of $ref^{\mathfrak{G}}$. We set

$$\begin{tikzcd} & \mathcal{M}^{\mathfrak{G}} \arrow[dd, dashed, "! f"] \\ \mathfrak{G} \arrow[ur, "ref^{\mathfrak{G}}"] \arrow[dr, "ref"'] & \\ & \mathcal{M} \end{tikzcd}$$

$$f := ref \circ (ref^{\mathfrak{G}})^{-1} \;\dot{\cup}\; \{(\lambda_{\mathcal{G}}^{\mathfrak{G}}(G), \lambda(G)) \mid \neg\exists v \in V.\, \rho(v) = G\} \; .$$

It is easy to see that f is a function from $G_0^{\mathfrak{G}}$ to G_0. We have to check 1)–3) of Def. 13.6.

1. If $G \in \mathcal{G}$ such that there is no $v \in V$ with $\rho(v) = G$, then 1) is fulfilled by definition of f. So we assume that there is a $v \in V$ with $\rho(v) = G$,

hence $ref^{\mathfrak{G}}(v) = \lambda_{\mathcal{G}}^{\mathfrak{G}}(G)$. Let $v_G := (ref^{\mathfrak{G}})^{-1}(\lambda_{\mathcal{G}}^{\mathfrak{G}}(G))$. We have $v\theta_{\mathfrak{G}}v_G$, hence $ref(v_G) \overset{\text{L. 13.3}}{=} ref(v) \overset{\mathcal{M}\models\mathfrak{G}[ref]}{=} \lambda_{\mathcal{G}}(G)$ We conclude

$$f(\lambda_{\mathcal{G}}^{\mathfrak{G}}(G)) = ref((ref^{\mathfrak{G}})^{-1}(\lambda_{\mathcal{G}}^{\mathfrak{G}}(G))) = ref(v_G) = \lambda_{\mathcal{G}}(G) \ ,$$

thus 1) is fulfilled.

2. Let $C \in \mathcal{C}$ and $g \in G_0^{\mathfrak{G}}$ with $g \in \mathrm{Ext}(\lambda_{\mathcal{C}}^{\mathfrak{G}}(C)) = \mathrm{Ext}(\mu(C))$. If $C = \top$, then 2) is fulfilled by definition of f. So we assume $C < \top$. Then we have a $v \in V$ with $g = [v]\theta_{\mathfrak{G}}$ and $\kappa(v) \leq C$. We set $v_g := (ref^{\mathfrak{G}})^{-1}(g)$, thus we have $v\theta_{\mathfrak{G}}v_g$. Similar to 1), we have $ref(v_g) = ref(v)$. This yields

$$f(g) = ref(ref^{\mathfrak{G}})^{-1}(g)) = ref(v_g) = ref(v) \in \mathrm{Ext}(\lambda_{\mathcal{C}}(\kappa(v))) \ .$$

From $\kappa(v) \leq C$, hence $\mathrm{Ext}(\lambda_{\mathcal{C}}(\kappa(v))) \subseteq \mathrm{Ext}(\lambda_{\mathcal{C}}(C))$, we conclude 2).
3. Is shown analogously. □

From the main theorem, we get the following corollary. The first equivalence of the corollary corresponds to Thm. 4.2.6. in [44]. But in [44], we find no result which can be compared to $\mathfrak{G}_1 \models \mathfrak{G}_2 \Leftrightarrow \mathcal{M}^{\mathfrak{G}_1} \models \mathcal{M}^{\mathfrak{G}_2}$. Again due to the lack of the concept of semantical entailment between models, Prediger has only proven an implication which is a weak variant of the implication $\mathfrak{G}_1 \models \mathfrak{G}_2 \Rightarrow \mathcal{M}^{\mathfrak{G}_1} \models \mathcal{M}^{\mathfrak{G}_2}$

Corollary 13.8.

Let $\mathfrak{G}_1, \mathfrak{G}_2$ be two graphs. Then we have

$$\mathfrak{G}_1 \models \mathfrak{G}_2 \quad \Longleftrightarrow \quad \mathcal{M}^{\mathfrak{G}_1} \models \mathfrak{G}_2 \quad \Longleftrightarrow \quad \mathcal{M}^{\mathfrak{G}_1} \models \mathcal{M}^{\mathfrak{G}_2} \ .$$

Proof: The direction '⇒' of the first equivalence is trivial, and the second equivalence follows immediately from Thm. 13.7. So it remains to show that '⇐' of the first equivalence holds.

So let $\mathcal{M} = (\vec{\mathbb{K}}, \lambda)$ be a contextual structure with $\mathcal{M} \models \mathfrak{G}_1$. Thm. 13.7 yields a mapping $f : G_0^{\mathfrak{G}_1} \to G_0$ such that $\mathcal{M} \models_f \mathcal{M}^{\mathfrak{G}_1}$ holds. We furthermore have a valuation $ref_1 : V \ \to G_1$ with $\mathcal{M}^{\mathfrak{G}_1} \models \mathfrak{G}_2[ref_1]$. We set $ref := f \circ ref_1$ and want to show that ref is a valuation with $\mathcal{M} \models \mathfrak{G}_2[ref]$.

$$\begin{tikzcd}
 & \mathcal{M}^{\mathfrak{G}_1} \arrow[dd, "f"] \\
\mathfrak{G}_2 \arrow[ur, "ref_1"] \arrow[dr, "!\, ref"'] & \\
 & \mathcal{M}
\end{tikzcd}$$

We have to check the vertex-conditions for ref, so let $v \in V_2$ be a vertex. Let $C := \kappa(v)$. As ref_1 fulfills the vertex-condition, we get $ref_1(v) \in \mathrm{Ext}(\lambda^{\mathfrak{G}_1}(C))$. Now condition 2) for f yields $ref(v) = f(ref_1(v)) \in \mathrm{Ext}(\lambda(C))$, which is the vertex-condition for v.

The edge-conditions are checked analogously. □

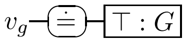
13.3 Standard Graphs

In the last section, we assigned to each graph a corresponding standard model which contains the same information as the graph. In this section, we do the same for the opposite direction: We assign to a model a standard graph which contains the same information. This is done with the following definition.

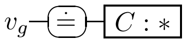
Definition 13.9 (Standard Graphs).

Let $\mathcal{M} := (\vec{\mathbb{K}}, \lambda)$ *be a contextual structure. We define the* standard graph *of* $\mathcal{M}$ *as follows:*

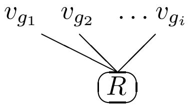
1. *For each* $g \in G_0$*, let* $v_g := \boxed{\top : *}$ *be a new vertex (i. e. we set* $\kappa(v_g) := \top$ *and* $\rho(v_g) = *$*).*
2. *For each* $g \in \mathcal{G}$ *with* $\lambda_{\mathcal{G}}(G) = g \in G_0$*, let* $v_{g,G}$ *be a new vertex and* $e_{g,G}$ *a new edge. Let* $\kappa(v_{g,G}) := \top$*,* $\rho(v_{g,G}) := G$*,* $\nu(e) := (v_g, v_{g,G})$ *and* $\kappa(e) :=\doteq$ *(i. e. we add the vertex and edge on the right).*



3. *For each* $C \in \mathcal{C}\backslash\{\top\}$ *with* $g \in Ext(\lambda_{\mathcal{C}}(C))$*, let* $v_{g,C}$ *be a new vertex and* $e_{g,C}$ *a new edge. Let* $\kappa(v_{g,C}) := C$*,* $\rho(v_{g,C}) := *$*,* $\nu(e) := (v_g, v_{g,C})$ *and* $\kappa(e) :=\doteq$ *(i. e. we add the vertex and edge on the right).*



4. *For each* $R \in \mathcal{R}_i\backslash\{\doteq\}$ *with* $(g_1, \ldots, g_i) \in Ext(\lambda_{\mathcal{R}}(R))$*, let* $e_{g_1,\ldots,g_i,R}$ *be a new vertex. Let* $\kappa(e_{g_1,\ldots,g_i,R}) := R$ *and* $\nu(e) := (g_1, \ldots, g_i)$ *(i. e. we add the edge on the right).*

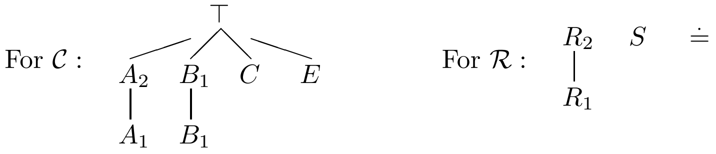


We denote this graph by $\mathfrak{G}_{(\vec{\mathbb{K}},\lambda)}$ *or* $\mathfrak{G}_{\mathcal{M}}$.[8]

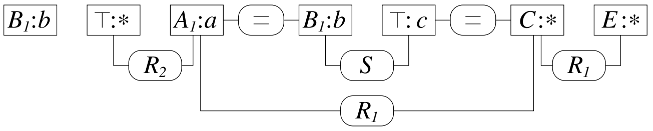
In [44], Def. 4.2.15, we find a corresponding definition for Def. 13.9. But there is a crucial difference between Prediger's approach and our approach: In [44], Prediger assigns a standard graph to a power context family instead to a contextual structure. Thus, she has first to define an alphabet which is derived from the power context family, then she defines the standard graph over this alphabet. Our approach is different: We fix an alphabet at the beginning and consider only graphs and structures over this fixed alphabet.

[8] $\mathfrak{G}_{\mathcal{M}}$ is given only up to isomorphism, but we have the implicit agreement that isomorphic graphs are identified.

To get an impression of standard models and standard graphs, we provide a more extensive example. First we have to fix the alphabet. We set $\mathcal{A} := (\{a, b, c, d\}, \{A_1, A_2, B_1, B_2, C, E, \top\}, \{R_1, R_2, S, \doteq\})$, where R_1, R_2, S are dyadic relation names. The orderings on the concept- and relation-names shall be as follows:



We consider the following graph over $\mathcal{A}$:

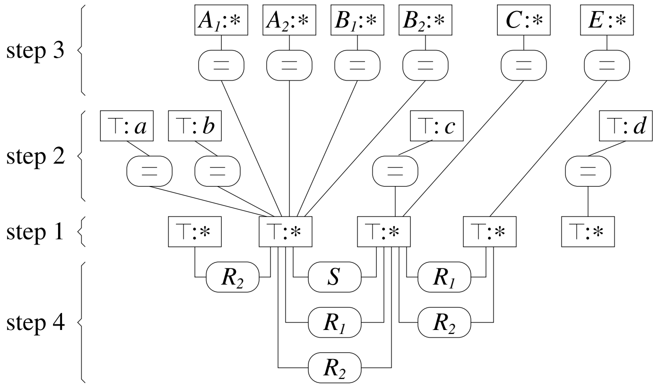


Below, we provide the standard model of this graph (the mappings λ_G, λ_C and λ_R are not explicit given, as they can easily be obtained from the power context family). We assume that the vertices of the graph are numbered from the left to the right, starting with 1, thus the i-th vertex is denoted by v_i.

$\mathbb{K}_0$	A_1	A_2	B_1	B_2	C	E	$\top$
$\{a, b, v_1, v_3, v_4\}$	×	×	×	×			×
$\{v_2\}$							×
$\{c, v_5, v_6\}$					×		×
$\{v_7\}$						×	×
$\{d\}$							×

$\mathbb{K}_2$	R_1	R_2	S	$\doteq$
$(\{v_2\}, \{a, b, v_1, v_3, v_4\})$		×		
$(\{a, b, v_1, v_3, v_4\}, \{c, v_5, v_6\})$	×	×	×	
$(\{c, v_5, v_6\}, \{v_7\})$	×	×		
$(\{v_2\}, \{v_2\})$				×
$(\{a, b, v_1, v_3, v_4\}, \{a, b, v_1, v_3, v_4\})$				×
$(\{c, v_5, v_6\}, \{c, v_5, v_6\})$				×
$(\{v_7\}, \{v_7\})$				×
$(\{d\}, \{d\})$				×

The standard graph of this model is given below. In the left, we sketch which vertices and edges are added by which step of Def. 13.9.



If we translate a model to a graph and then translate this graph back into a model, in general we do not get the same graph back, but at least a semantically equivalent graph:

Lemma 13.10 ($\mathcal{M}$ and $\mathcal{M}^{\mathfrak{G}_\mathcal{M}}$ are Semantically Equivalent).

Let $\mathcal{A} = (\mathcal{G}, \mathcal{C}, \mathcal{R})$ be an alphabet and let $\mathcal{M}$ be a contextual structure over $\mathcal{A}$.

1. *It holds $\mathcal{M} \models \mathcal{M}^{\mathfrak{G}_\mathcal{M}}$ and $\mathcal{M}^{\mathfrak{G}_\mathcal{M}} \models \mathcal{M}$.*
2. *If $\mathcal{M}$ satisfies furthermore $M_0 = \mathcal{C}$ and $M_i = \mathcal{R}_i$ for all $i \geq 1$, then $\mathcal{M}$ and $\mathcal{M}^{\mathfrak{G}_\mathcal{M}}$ are even isomorphic.*

Proof: With the denotation of Def. 13.9 for v_g, let $f := \begin{cases} \mathcal{M} \to \mathcal{M}^{\mathfrak{G}_\mathcal{M}} \\ \quad g \mapsto [v_g]\theta_{\mathfrak{G}_\mathcal{M}} \end{cases}$.

It can easily be checked that f is bijective, and that we have $\mathcal{M} \models_{f^{-1}} \mathcal{M}^{\mathfrak{G}_\mathcal{M}}$ and $\mathcal{M}^{\mathfrak{G}_\mathcal{M}} \models_f \mathcal{M}$, hence 1 is fulfilled. For 2, we have that for each i, the contexts $\mathbb{K}_i$ and $\mathbb{K}_i^{\mathfrak{G}_\mathcal{M}}$ have the same attributes, so the conditions which are satisfied by f yield that $\mathcal{M}$ and $\mathcal{M}^{\mathfrak{G}_\mathcal{M}}$ are isomorphic. □

From 2. of Lem. 13.10 we conclude that each model $\mathcal{M}$ with $M_0 = \mathcal{C}$ and $M_i = \mathcal{R}_i$ for all $i \geq 1$ is already isomorphic to a standard model of a graph. Together with 1., we see that each class in the quasiorder (CS,$\models$) contains at least one standard model (but this is not uniquely determined: Each class contains infinitely many pairwise non-isomorphic standard models).

In the following, we will provide an corresponding result of the last lemma for graphs, that is we will show that $\mathfrak{G}$ and $\mathfrak{G}_{\mathcal{M}^\mathfrak{G}}$ are equivalent. In contrast to the last lemma, we will prove that $\mathfrak{G}$ and $\mathfrak{G}_{\mathcal{M}^\mathfrak{G}}$ are *syntactically* equivalent. As we know from [15] that the calculus for concept graphs with cuts is sound, we know that the restricted calculus $\vdash_{pos}$ we consider in this paper is sound, too. In particular, when we have shown that $\mathfrak{G}$ and $\mathfrak{G}_{\mathcal{M}^\mathfrak{G}}$ are syntactically equivalent, we know that are these graphs are semantically equivalent as well. Before we prove this equivalence, we need a simple lemma.

Lemma 13.11.

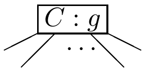
Let $\mathfrak{G} := (V, E, \nu, \kappa, \rho)$ be a concept graph, let v_1, v_2 be two new vertices, e_1, e_2 be two new edges, and let $\mathfrak{G}' := (V', E', \nu', \kappa', \rho')$ be defined as follows:

- $V' := V \dot\cup \{v_1, v_2\}$, $E' := E \dot\cup \{e_1, e_2\}$,
- $\nu' := \nu \dot\cup \{(e_1, (v, v_1)), (e_2, (v, v_2))\}$
- $\kappa' := \kappa|_{V\backslash\{v\}\dot\cup E} \dot\cup \{(v, \top), (v_1, \top), (v_2, \kappa(v)), (e_1, \doteq), (e_2, \doteq)\}$, *and*
- $\rho' := \rho|_{V\backslash\{v\}} \dot\cup \{(v, *), (v_1, \rho(v)), (v_2, *)\}$.

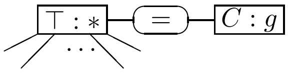
Then we have $\mathfrak{G} \vdash_{pos} \mathfrak{G}'$ *and* $\mathfrak{G}' \vdash_{pos} \mathfrak{G}$.

Proof:

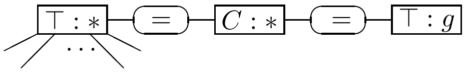
We start with:



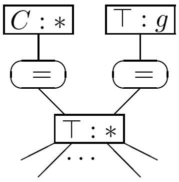
Cor. 10.4. yields:



Lem. 10.3 yields:



Lem. 10.5 yields:



□

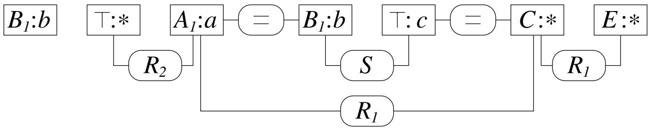
Now we are prepared to prove the syntactical equivalence of $\mathfrak{G}$ and $\mathfrak{G}_{\mathcal{M}^{\mathfrak{G}}}$.

Theorem 13.12 ($\mathfrak{G}$ and $\mathfrak{G}_{\mathcal{M}^{\mathfrak{G}}}$ are Syntactically Equivalent).

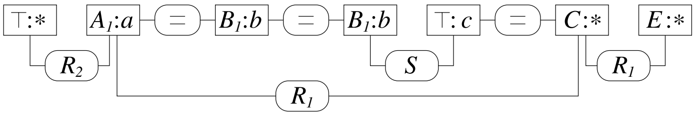
Let $\mathfrak{G}$ be a concept graph. Then we have

$$\mathfrak{G} \vdash_{pos} \mathfrak{G}_{\mathcal{M}^{\mathfrak{G}}} \quad \textit{and} \quad \mathfrak{G}_{\mathcal{M}^{\mathfrak{G}}} \vdash_{pos} \mathfrak{G} .$$

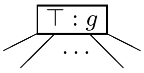
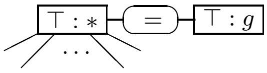
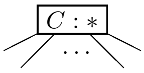
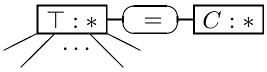
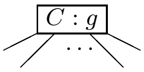
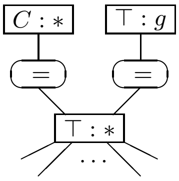
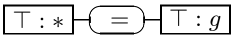
Proof: We will exemplify the proof with the example for standard graphs above, that is, we start with



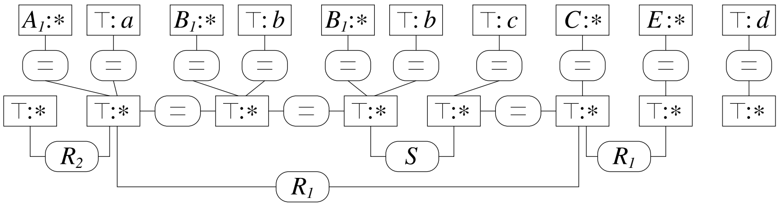
Let $[v]\theta_{\mathfrak{G}} \cap V = \{v_1, \ldots, v_n\}$. With the rule 'identity-insertion' and with Lem. 11.5 of [15], we can add or remove identity links such that there is an identity link between $v_i, v_j \in V \cap [v]\theta_{\mathfrak{G}}$ iff $j = i + 1$. One possible result for our example is:



Now we do the following:

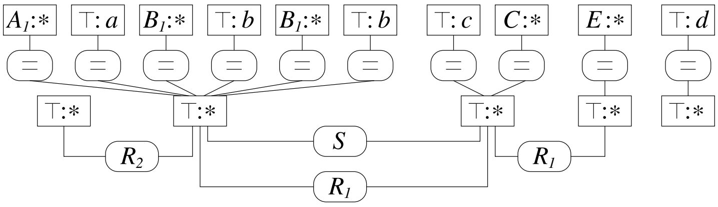
1. Each vertex  is replaced by 
2. Each vertex  $(C \neq \top)$ is replaced by 
3. Each vertex  is replaced with Lem. 13.11 by 
4. For each $G \in \mathcal{G}$ with $g \notin \rho[V]$ we add 

For our example, we get:

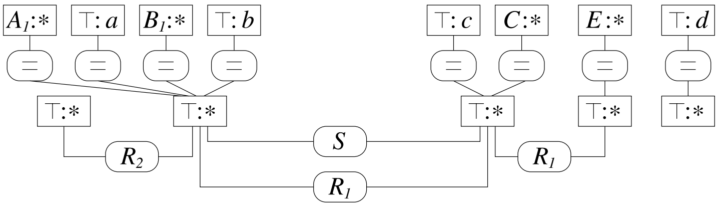


For each class $[v]\theta_{\mathfrak{G}} := \{v_1, \ldots, v_n\}$, we merge v_1 into v_2, v_2 into v_3, …, v_{n-1} into v_n. After this step, each class $[v]\theta_{\mathfrak{G}}$ corresponds to exactly one vertex in the graph we have constructed so far. Let W be the set of these vertices.

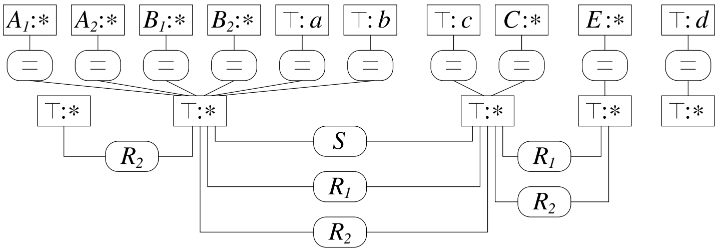
In our example, we get:



Now we erase all repeated instances of structures $-(\;=\;)-\boxed{C : *}$ and of structures $-(\;=\;)-\boxed{\top : g}$ which are linked to the same vertex $w = \boxed{\top : *}$ with $w \in W$. Analogously, all repeated instances of edges are erased. The opposite direction can be carried out with the iteration-rule.



For each vertex $w \in W$, we now do the following: If a structure $-(\;=\;)-\boxed{C : g}$ is linked to w and if $C < D < \top$, we add a structure $-(\;=\;)-\boxed{D : g}$ which is linked to w as well (supposed the new structure did not exist already). Analogously, if we have an edge e which is incident with vertices from W and which is labelled with the relation name $R := \kappa(e)$, and if S is an relation name with $R < S$, we add an edge f with $\kappa(f) = S$, $|e| = |f| = k$ and $f|_1 = e|_1, \ldots, f|_k = e|_k$ — supposed an edge like this did not exists already. This can be done with an application the iteration- and of the generalization-rule. The opposite direction can be carried out with the erasure-rule.



It is easy to see that the resulting graph is $\mathfrak{G}_{\mathcal{M}^{\mathfrak{G}}}$. As all steps in the proof can be carried out in both directions, we are done. □

The class of all graphs over a given alphabet $\mathcal{A}$, together with the semantical entailment relation $\models$ is a quasiorder. The same holds for the class of all models. With the last theorem, we are now prepared to show that these quasiorders are isomorphic structures. More precisely, we have the following corollary:

Corollary 13.13 ((CS,$\models$), (CG,$\models$) are Isomorphic Quasiorders).

The mappings $\mathcal{M} \mapsto \mathfrak{G}_{\mathcal{M}}$ and $\mathfrak{G} \mapsto \mathcal{M}^{\mathfrak{G}}$ are, up to equivalence, mutually inverse isomorphisms between the quasiordered sets (CS , $\models$) and (CG , $\models$).

Proof: As we know that $\vdash_{pos}$ is sound, the last theorem yields that $\mathfrak{G} \models \mathfrak{G}_{\mathcal{M}^{\mathfrak{G}}}$ and $\mathfrak{G}_{\mathcal{M}^{\mathfrak{G}}} \models \mathfrak{G}$ hold as well. We have furthermore

$$\mathcal{M}_1 \models \mathcal{M}_2 \overset{\text{Lem. 13.10}}{\Longleftrightarrow} \mathcal{M}^{\mathfrak{G}_{\mathcal{M}_1}} \models \mathcal{M}^{\mathfrak{G}_{\mathcal{M}_2}} \overset{\text{Cor. 13.8}}{\Longleftrightarrow} \mathfrak{G}_{\mathcal{M}_1} \models \mathfrak{G}_{\mathcal{M}_2} \; .$$

These results together with Lem. 13.10 and Cor. 13.8 yield this corollary. □

13.4 Transformation-Rules for Models

We still have to show that $\vdash_{pos}$ is complete. Although we have the last corollary, this cannot be derived from the results we have so far.

In order to prove the completeness of $\vdash_{pos}$, we will introduce four transformation rules for models:[9] *removing an element, doubling an element, exchanging attributes, and restricting the incidence relations.* This rules form a calculus for models, which will be denoted by $\vdash$. We will show that $\vdash_{pos}$ for graphs and $\vdash$ for models are complete. The main idea is the following: If we have two models $\mathcal{M}^a$ and $\mathcal{M}^b$ with $\mathcal{M}^a \models \mathcal{M}^b$, we will show that $\mathcal{M}^a$ can successively transformed to $\mathcal{M}^b$ with the rules for the models, and each transformation carries over to the standard graphs, i. e. we get simultaneous $\mathfrak{G}_{\mathcal{M}^a} \vdash_{pos} \mathfrak{G}_{\mathcal{M}^b}$. This is enough to prove the completeness of $\vdash_{pos}$ as well.

In the following, we will define the transformation rules for models and show that each rule is sound in the system of models, and that it carries over to the set of graphs, together with the calculus $\vdash_{pos}$.

Definition 13.14 (Removing an Element).

Let $\mathcal{M} := (\vec{\mathbb{K}}, \lambda)$ be a contextual $\mathcal{A}$-structure and let $g \in G_0 \backslash \lambda_{\mathcal{G}}[\mathcal{G}]$. We define a power context family $\vec{\mathbb{K}}'$ as follows:

[9] We have the implicit agreement that isomorphic models are identified. Isomorphism between power context families and between models is defined as usual.

1. $G'_0 = G_0 \backslash \{g\}$, $G'_i = G_i \cap (G'_0)^i$ *for all* $i \geq 1$,
2. $M'_i = M_i$ *for all* i,
3. $I'_i = I_i \cap (G'_i \times M_i)$ *for all* i.

For the contextual structure $\mathcal{M} := (\vec{\mathbb{K}}', \lambda)$ *, we say that* $\mathcal{M}'$ *is obtained from* $\mathcal{M}$ *by* removing the element g.

Lemma 13.15 (Removing an Element).

Let $\mathcal{M} := (\vec{\mathbb{K}}, \lambda)$ *be a model, let* $\mathcal{M}' := (\vec{\mathbb{K}}', \lambda')$ *be obtained from* $\mathcal{M}$ *by removing the element* g*. Let* $\mathcal{M}^a := (\vec{\mathbb{K}}^a, \lambda^a)$ *be a model, let* $f : G_0^a \to G_0$ *with* $g \notin f[G_0^a]$ *and* $\mathcal{M} \models_f \mathcal{M}_a$*, and* $id : G_0 \backslash \{g\} \to G_0$ *the identity-mapping. Then we have*

$$\mathcal{M}' \models_f \mathcal{M}^a \quad , \quad \mathcal{M} \models_{id} \mathcal{M}' \quad \text{and} \quad \mathfrak{G}_{\mathcal{M}} \vdash_{pos} \mathfrak{G}_{\mathcal{M}'}$$

Proof: It is easy to check that we have $\mathcal{M}' \models_f \mathcal{M}^a$ and $\mathcal{M} \models_{id} \mathcal{M}'$.

With the denotations from Def. 13.9), $\mathfrak{G}_{\mathcal{M}'}$ can be derived from $\mathfrak{G}_{\mathcal{M}}$ by easing all edges v_g is incident with, by erasing all vertices and edges $v_{g,G}$ and $e_{g,G}$ with $G \in \mathcal{G}$ and $\lambda_{\mathcal{G}}(G) = g$, by erasing all vertices and edges $v_{g,C}$ and $e_{g,C}$ with $C \in \mathcal{C}$ and $g \in \text{Ext}(\lambda_{\mathcal{C}}(C))$, and by erasing v_g. □

Definition 13.16 (Doubling an Element).

Let $\mathcal{M} := (\vec{\mathbb{K}}, \lambda)$ *be a contextual* $\mathcal{A}$*-structure and let* $g, g' \in G_0$*. For each tuple* $\vec{h} = (g_1, \ldots, g_i) \in G_i$*, we set* $\vec{h}[g'/g] := (g'_1, \ldots, g'_i)$ *with* $g'_j := \begin{cases} g_j & g_j \neq g \\ g' & g_j = g \end{cases}$.

If $\mathcal{M}' := (\vec{\mathbb{K}}', \lambda')$ *is a contextual structure over* $\mathcal{A}$ *such that there is a* $g \in G_0$ *with*

1. $G'_0 = G_0 \mathbin{\dot\cup} \{g'\}$ *for a* $g' \notin G_0$ *and* $G'_i = G_i \mathbin{\dot\cup} \{\vec{h}[g'/g] \mid \vec{h} \in G_i\}$ *for all* $i \geq 1$,
2. $M'_i = M_i$ *for all* i,
3. $I'_0 = I_0 \mathbin{\dot\cup} \{(g', m) \mid gI_0m\}$*, and* $I'_i = I_i \mathbin{\dot\cup} \{(\vec{h}[g'/g], m) \mid \vec{h}I_im\}$ *for all* $i \geq 1$,
4. $\lambda'_{\mathcal{G}}$ *fulfills* $\lambda'_{\mathcal{G}}(G) = \lambda_{\mathcal{G}}(G)$ *for all* $G \in \mathcal{G}$ *with* $\lambda_{\mathcal{G}}(G) \neq G$*, and for all* $G \in \mathcal{G}$ *with* $\lambda_{\mathcal{G}}(G) = G$ *we have* $\lambda'_{\mathcal{G}}(G) \in \{g, g'\}$,
5. $\lambda'_{\mathcal{C}}(C) = (\,(\mathit{Int}(\lambda_{\mathcal{C}}(C))^{I'_0}, \mathit{Int}(\lambda_{\mathcal{C}}(C))\,)$ *for all* $C \in \mathcal{C}$*, and*
6. $\lambda'_{\mathcal{R}}(R) = (\,(\mathit{Int}(\lambda_{\mathcal{R}}(R))^{I'_i}, \mathit{Int}(\lambda_{\mathcal{R}}(R))\,)$ *for all* $R \in \mathcal{R}_i$,

then we say that $\mathcal{M}'$ is obtained from $\mathcal{M}$ by doubling the element g.

As the definition of *doubling an element* is fairly technical, we provide an example for this rule. Let $\mathcal{A} := (\{a,b,c\},\{A,B,\top\},\{R,\dot{=}\})$, where R is a 4-ary relation name, and let the following contextual structure $\mathcal{M}$ over $\mathcal{A}$ be given (we have added an additional column to show how the mapping $\lambda_{\mathcal{G}}$ assigns object names to objects):

$\lambda_{\mathcal{G}}$
a,b
c

$\mathbb{K}_0$	A	B	C	$\top$
g	×	×		×
h			×	×

$\mathbb{K}_2$	$\dot{=}$
(g,g)	×
(h,h)	×

$\mathbb{K}_4$	R
(g,h,g,h)	×

.

Then the following contextual structure is one of three possible structures which can be obtained from $\mathcal{M}$ by doubling the element g:

$\lambda_{\mathcal{G}}$
a
b
c

$\mathbb{K}_0$	A	B	C	$\top$
g	×	×		×
g'	×	×		×
h			×	×

$\mathbb{K}_2$	$\dot{=}$
(g,g)	×
(g',g')	×
(h,h)	×

$\mathbb{K}_4$	R
(g,h,g,h)	×
(g',h,g',h)	×

.

As we loose the information $\lambda_{\mathcal{G}}(a) = \lambda_{\mathcal{G}}(b)$, we see that this contextual structure contains less information than $\mathcal{M}$.

Furthermore we note that if $(\vec{\mathbb{K}}',\lambda'),(\vec{\mathbb{K}}'',\lambda'')$ are two contextual structures which are obtained from a contextual structure $(\vec{\mathbb{K}},\lambda)$ by doubling the element $g \in G_0$, then they can (up to isomorphism) only differ between $\lambda'_{\mathcal{G}}$ and $\lambda''_{\mathcal{G}}$.

Lemma 13.17 (Doubling an Element).

Let $\mathcal{M} := (\vec{\mathbb{K}},\lambda)$ be a model and let $\mathcal{M}' := (\vec{\mathbb{K}}',\lambda')$ be obtained from $\mathcal{M}$ by doubling the element $g \in G_0$. Then we have

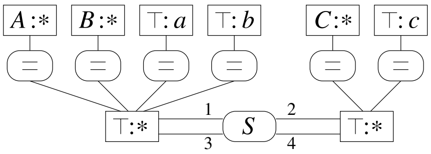
$$\mathcal{M} \models \mathcal{M}' \quad \textit{and} \quad \mathfrak{G}_{\mathcal{M}} \vdash_{pos} \mathfrak{G}_{\mathcal{M}'} \ .$$

If we have a model $\mathcal{M}^a := (\vec{\mathbb{K}}^a,\lambda^a)$ with $\mathcal{M} \models \mathcal{M}_a$, then we can chose $\mathcal{M}'$ such that we have $\mathcal{M}' \models \mathcal{M}^a$.

Proof: Let $g' \in G'_0$ be the new element which is obtained from doubling $g \in G_0$, i. e. $G'_0 = G_0 \dot{\cup} \{g'\}$. It is easy to see that $f' : G'_0 \to G_0$ with $f'|_{G_0} = id$ and $f'(g') = g$ fulfills all conditions of Def. 13.6, i. e. we have $\mathcal{M} \models_{f'} \mathcal{M}'$.

Now let $f_a : G^a_0 \to G_0$ be a mapping with $\mathcal{M} \models_f \mathcal{M}^a$. Let f'_a be an arbitrary mapping with $f'_a(h) = f_a(h)$, if $f_a(h) \neq g$, and $f'_a(h) \in \{g,g'\}$, if $f_a(h) = g$. Now condition 4. of Def. 13.16 enables us to chose $\lambda'_{\mathcal{G}}$ from $\mathcal{M}'$ such that condition 1. of Def. 13.6 is satisfied. Conditions 2. and 3. of Def. 13.6 are trivially satisfied. So we have $\mathcal{M}' \models_{f'_a} \mathcal{M}^a$.

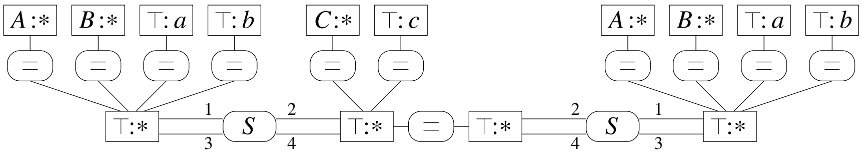
It remains to show that $\mathfrak{G}_{\mathcal{M}} \vdash_{pos} \mathfrak{G}_{\mathcal{M}'}$ holds. We will exemplify the proof with the example after Def. 13.16, that is, we start with



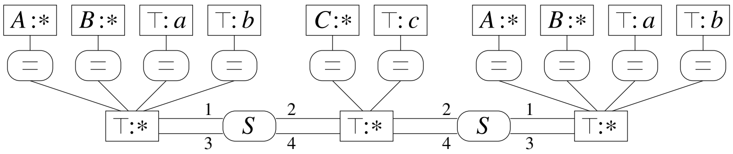
We consider the subgraph which contains the following vertices and edges (we use the denotation of Def. 13.9):

1. v_g, all $v_{g,G}$ and $e_{g,G}$ (with $G \in \mathcal{G}$ and $\lambda_{\mathcal{G}}(G) = g$), all $v_{g,C}$ and $e_{g,C}$ (with $C \in \mathcal{C}$ and $g \in \text{Ext}(\lambda_{\mathcal{C}}(C))$.
2. All edges $e_{g_1,\ldots,g_i,R}$ such that $g_j = g$ for one j (with $R \in \mathcal{R}$ and $(g_1, \ldots, g_i) \in \text{Ext}(\lambda_{\mathcal{R}}(R)))$. The set of these edges shall be denoted by F.
3. All vertices v_h which are incident with an edge $e \in F$. The set of these vertices shall be denoted by W.

This subgraph is iterated, and an new identity link is inserted between w and its copy for each $w \in W$.[10] The copies of the vertices $v_{g,G}$ and edges $e_{g,G}$ will be denoted by $v'_{g,G}$ and $e'_{g,G}$, respectively. For our example, we get:

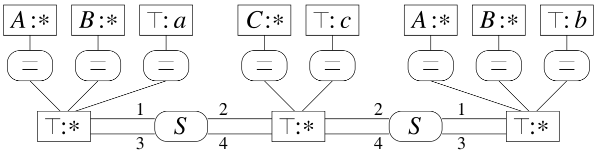


For every $w \in W$, the copy of w is merged 'back' into w. For our example, we get:



For g, we erase all vertices $v_{g,G}$ and edges $e_{g,G}$, where $\lambda'_{\mathcal{G}}(G) \neq g$. Analogously for g', we erase all vertices $v'_{g,G}$ and edges $e'_{g,G}$, where $\lambda'_{\mathcal{G}}(G) \neq g'$. For our example, we obtain the following graph:

[10] Using the technical implementation of the iteration-rule in Def. 11.1. of [15], we insert an identity link between $(w, 1)$ and $(w, 2)$.



This is $\mathfrak{G}_{\mathcal{M}'}$. □

Definition 13.18 (Restricting the Incidence Relations).

Let $\mathcal{M} := (\vec{\mathbb{K}}, \lambda)$ be a contextual $\mathcal{A}$-structure. If $\mathcal{M}' := (\vec{\mathbb{K}}', \lambda)$ is a contextual structure over $\mathcal{A}$ with

$$G'_i = G_i \text{ for all } i, \quad M'_i = M_i \text{ for all } i, \quad \text{and } I'_i \subseteq I_i \text{ for all } i \ ,$$

then we say that $\mathcal{M}'$ is obtained from $\mathcal{M}$ by restricting the incidence relations*.*

Lemma 13.19 (Restricting the Incidence Relations).

If $\mathcal{M}' := (\vec{\mathbb{K}}', \lambda)$ is obtained from $\mathcal{M} := (\vec{\mathbb{K}}, \lambda)$ by restriction the incidence relations, we have

$$\mathcal{M} \models_{id} \mathcal{M}' \quad \text{and} \quad \mathfrak{G}_{\mathcal{M}} \vdash_{pos} \mathfrak{G}_{\mathcal{M}'} \ .$$

Proof: It is easy to see that $\mathfrak{G}_{\mathcal{M}'}$ is a subgraph of $\mathfrak{G}_{\mathcal{M}}$, hence $\mathfrak{G}_{\mathcal{M}'}$ can be derived from $\mathfrak{G}_{\mathcal{M}}$ by erasing all edges and vertices which are not in $\mathfrak{G}_{\mathcal{M}'}$. □

Definition 13.20 (Exchanging Attributes and Standardization).

Let $\mathcal{M} := (\vec{\mathbb{K}}, \lambda)$ be a contextual $\mathcal{A}$-structure. If $\mathcal{M}' := (\vec{\mathbb{K}}', \lambda')$ is a contextual $\mathcal{A}$-structure which satisfies

1. *$G'_i := G_i$ for all i,*
2. *$gI_0\lambda_{\mathcal{C}}(C) \Longleftrightarrow gI'_0\lambda'_{\mathcal{C}}(C)$ for all $g \in G_0$ and $C \in \mathcal{C}$,*
3. *$\vec{g}I_i\lambda_{\mathcal{R}}(R) \Longleftrightarrow \vec{g}I'_i\lambda'_{\mathcal{C}}(R)$, for all $i \geq 1$, $\vec{g} \in G_i$ and $R \in \mathcal{R}_i$, and*
4. *$\lambda'_{\mathcal{G}} := \lambda_{\mathcal{G}}$,*

then we say that $\mathcal{M}'$ is obtained from $\mathcal{M}$ by exchanging attributes of $\mathcal{M}$*. If $\mathcal{M}'$ additionally satisfies $M'_0 := \mathcal{C}$ and $M'_i := \mathcal{R}_i$ for all $i \geq 1$, then we say that $\mathcal{M}'$ is obtained from $\mathcal{M}$ by* standardization of $\mathcal{M}$*.*

This rule is the only rule which does not weaken the informational content of a model. In particular, it can be carried out in both directions.

Lemma 13.21 (Exchanging Attributes and Standardization).

If $\mathcal{M}' := (\vec{\mathbb{K}}', \lambda')$ is obtained from $\mathcal{M} := (\vec{\mathbb{K}}, \lambda)$ by exchanging attributes, we have

$$\mathcal{M} \models_{id} \mathcal{M}' \quad , \quad \mathcal{M}' \models_{id} \mathcal{M} \quad , \text{ and } \quad \mathfrak{G}_{\mathcal{M}} = \mathfrak{G}_{\mathcal{M}'} \ .$$

Furthermore exists a standardization of $\mathcal{M}$ for each contextual structure $\mathcal{M}$.

Proof: Trivial

The four rules form a calculus for models, i. e. we have the following definition:

Definition 13.22 (Calculus for Contextual Structures).

The calculus for contextual structures over the alphabet $\mathcal{A} := (\mathcal{G}, \mathcal{C}, \mathcal{R})$ consists of the following rules:

Removing an element, doubling an element, exchanging attributes, and restricting the incidence relations.

If $\mathcal{M}^a$, $\mathcal{M}^b$ are two models, and if there is a sequence $(\mathcal{M}^1, \mathcal{M}^2, \ldots, \mathcal{M}^n)$ with $\mathcal{M}^1 = \mathcal{M}^a$ and $\mathcal{M}^b = \mathcal{M}^n$ such that each $\mathcal{M}^{i+1}$ is derived from $\mathcal{M}^i$ by applying one of the rules of the calculus, we say that $\mathcal{M}^b$ can be derived from *$\mathcal{M}^a$, which is denoted by $\mathcal{M}^a \vdash \mathcal{M}^b$.*

Now we are prepared to show that the transformation rules for models are complete and respected by the construction of standard graphs.

Theorem 13.23 ($\vdash$ is Complete and Respected by Stand. Graphs).

Let $\mathcal{M}^a := (\vec{\mathbb{K}}^a, \lambda^a)$, $\mathcal{M}^b := (\vec{\mathbb{K}}^b, \lambda^b)$ be two contextual structures such that $\mathcal{M}^a \models \mathcal{M}^b$. Then we have

$$\mathcal{M}^a \vdash \mathcal{M}^b \quad \text{ and } \quad \mathfrak{G}_{\mathcal{M}^a} \vdash_{pos} \mathfrak{G}_{\mathcal{M}^b} \ .$$

Proof: First Lem. 13.21 allows us to assume w.l.o.g. that $\mathcal{M}^a$ and $\mathcal{M}^b$ are standardized, i. e. that we have $M_0^a = M_0^b = \mathcal{C}$ and $M_i^a = M_i^b = \mathcal{R}_i$ for all $i \geq 1$.

Let $f : G_0^b \to G_0^a$ with $\mathcal{M}^a \models_f \mathcal{M}^b$.

Assume that f is not injective, i. e. there are $g_1, g_2 \in G_0^b$ with $f(g_1) = f(g_2)$. Then we can double $f(g_1) \in G_0^a$ to obtain from $\mathcal{M}^a$ a contextual model $\mathcal{M}^c$ with a new element h. Similar to the proof of Lem. 13.17, we can choose $\lambda_{\mathcal{C}}^c$ such that the a mapping $f' : G_0^b \to G_0^c$ with $f'|_{G_0^b \backslash \{g_2\}} = f$ and $f(g_2) = h$ which fulfills $\mathcal{M}^c \models_{f'} \mathcal{M}$. If f' is not injective again, we repeat this step as often as necessary until we finally obtain a contextual structure $\mathcal{M}^1$ and an injective mapping $f_1 : G_0^b \to G_0^1$ with $\mathcal{M}^1 \models_{f_1} \mathcal{M}^b$. As $\mathcal{M}^1$ is obtained from $\mathcal{M}^a$ by doubling several elements, we have $\mathcal{M}^a \vdash \mathcal{M}^1$ by definition of $\vdash$, and Lem. 13.17 yields $\mathfrak{G}_{\mathcal{M}^a} \vdash_{pos} \mathfrak{G}_{\mathcal{M}^1}$.

If f_1 is not surjective, we can remove with the rule rule 'removing an element' gradually all objects $g \in G_0^1 \backslash f_1[G_0^b]$ from $\mathcal{M}^1$ to obtain from $\mathcal{M}^1$ a contextual structure $\mathcal{M}^2$ with $\mathcal{M}^1 \vdash \mathcal{M}^2$. Lemma 13.15 yields $\mathfrak{G}_{\mathcal{M}^1} \vdash_{pos} \mathfrak{G}_{\mathcal{M}^2}$ and $\mathcal{M}^2 \models_{f_1} \mathcal{M}^b$. Furthermore we now have that f_2 is bijective.

It is clear that isomorphic contextual structures yield isomorphic standard graphs, so we can finally assume that f is the identity-mapping, in particular we have $G_0^2 = G_0^b$ (remember that have the implicit agreement that isomorphic models are identified).

Conditions 1.–3. of Def. 13.6, which are satisfied by id, can now be stated as follows:

1. For all $G \in \mathcal{G}$ we have $\lambda_{\mathcal{G}}^b(G) = \lambda_{\mathcal{G}}^a(G)$,
2. $\mathcal{C} = {M_0}^2 = {M_0}^b$, and for all $C \in \mathcal{C}$ we have $C^{{I_0}^2} \subseteq C^{{I_0}^b}$, and
3. for $i \geq 1$, we have $\mathcal{R}_i = {M_i}^2 = {M_i}^b$, and for all $R \in \mathcal{R}_i$ we have $R^{{I_i}^2} \subseteq R^{{I_i}^b}$.

Now it is easy to see that $\mathcal{M}^b$ can be obtained from $\mathcal{M}^2$ by restricting the incidence relations, thus Lem. 13.19 yields $\mathcal{M}^2 \vdash \mathcal{M}^b$ and $\mathfrak{G}_{\mathcal{M}^2} \vdash_{pos} \mathfrak{G}_{\mathcal{M}^b}$.

As $\vdash_{pos}$ (for graphs) and $\vdash$ (for models) are transitive, we conclude $\mathcal{M}^a \vdash \mathcal{M}^b$ and $\mathfrak{G}_{\mathcal{M}^a} \vdash_{pos} \mathfrak{G}_{\mathcal{M}^b}$. □

From this theorem we can conclude that the calculus $\vdash_{pos}$ on the graphs is complete as well, as the following corollary shows.

Corollary 13.24 (Both Calculi are Complete).
Let $\mathcal{M}_1, \mathcal{M}_2$ be models and $\mathfrak{G}_1, \mathfrak{G}_2$ be graphs. We have:

$$\mathcal{M}_1 \models \mathcal{M}_2 \iff \mathcal{M}_1 \vdash \mathcal{M}_2 \quad \textit{and} \quad \mathfrak{G}_1 \models \mathfrak{G}_2 \iff \mathfrak{G}_1 \vdash_{pos} \mathfrak{G}_2 \ .$$

Proof: The direction '⇐' of the first equivalence follows immediately from Lemmata 13.17, 13.15, 13.21, and 13.19, and the direction '⇒' is a part of Thm. 13.23.

The direction '⇐' of the second equivalence is already proven in [15], so it remains to show '⇒'. We have:

$$\mathfrak{G}_1 \models \mathfrak{G}_2 \overset{\text{C. 13.8}}{\iff} \mathcal{M}^{\mathfrak{G}_1} \models \mathcal{M}^{\mathfrak{G}_2} \overset{\text{C. 13.13}}{\Longrightarrow} \mathfrak{G}_{\mathcal{M}^{\mathfrak{G}_1}} \models \mathfrak{G}_{\mathcal{M}^{\mathfrak{G}_2}} \overset{\text{T. 13.12}}{\iff} \mathfrak{G}_1 \models \mathfrak{G}_2 \ ,$$

thus we are done. □

13.5 Conclusion

In Prediger's work ([44]), the notion of standard models is adaquate only for concept graphs without generic markers. For concept graphs with generic markers, a standard model of a graph may encode less information than the graph. Thus, in this case, the reasoning on concept graphs cannot be carried over to the models completely. This is a gap we have bridged with our notion of standard models, which extends Prediger's approach. Moreover, reasoning on models can now be carried out in two different ways: By the semantical entailment relation $\models$ on models (see Def. 13.6), and by transformation rules between models (see Sect. 13.4).

In this tratise, an adaquate calculus for concept graphs with cuts is provided. Thus, if we have two concept graphs without cuts $\mathfrak{G}_a$ and $\mathfrak{G}_b$ with $\mathfrak{G}_a \models \mathfrak{G}_b$, we have a proof for $\mathfrak{G}_a \vdash \mathfrak{G}_b$, that is: We have a sequence $(\mathfrak{G}_1, \mathfrak{G}_2, \ldots, \mathfrak{G}_n)$ with $\mathfrak{G}_1 = \mathfrak{G}_a$, $\mathfrak{G}_b = \mathfrak{G}_n$ such that each $\mathfrak{G}_{i+1}$ is derived from $\mathfrak{G}_i$ by applying one of the rules of the calculus. But now we have even more: From Cor. 13.24, we conclude that we can find a proof $(\mathfrak{G}_1, \mathfrak{G}_2, \ldots, \mathfrak{G}_n)$ such that *all* graphs $\mathfrak{G}_1, \ldots, \mathfrak{G}_n$ are concept graphs without cuts. This result cannot be directly derived from the preceeding chapters of this treatise.

14 Design Decisions

Concept graphs are a mathematization of conceptual graphs. In the process of mathematization, the negation contexts of conceptual graphs were replaced by cuts, and the coreference links of conceptual graphs where replaced by identity links. It is important to note that this have been *decisions* in the process of mathematization. In this chapter, some arguments for these decisions are presented. Moreover, it will be argued why we restricted ourselves to concept graphs with dominating nodes, and the design of the calculus will be explained.

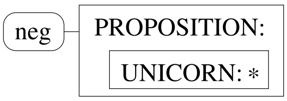
14.1 Cuts

Negations occur in several approaches for conceptual graphs. Why we did not adopt and mathematizise one of these approaches shall be explained in this section.

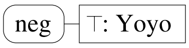
In order to handle negations, a specific syntactical element of the well-formed formulas has to be declared to express them. For this purpose, the standard approach uses a context box of type `Proposition` which is linked to a unary relation of type `neg` (see [64]). Sometimes, these special context boxes are abbreviated by context boxes of type `Negation` (e.g. [65]) or by drawing a simple rectangle with the mathematical negation symbol $\neg$ ([64]). Some approaches use these rectangles without explaining whether the box is an independent syntactical element or just an abbreviation for context boxes of a specific type (e.g. [70] or [1]). But as soon as negation is introduced, the definition of conceptual graphs (i.e., the definition of the well-formed formulas), any calculus and any translation of conceptual graphs to another formal language (like the translation to first order logic with the Φ-operator) have to capture its properties.

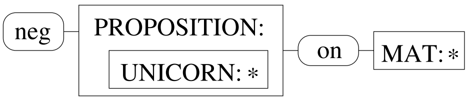
We start our discussion with the definition of conceptual graphs. In [64] we find that a context is a box whose designator is a concept graph. As boxes may be linked to relations, the following is a valid conceptual graph:

F. Dau: The Logic System of Concept Graphs with Negation, LNAI 2892, pp. 187-204, 2003.



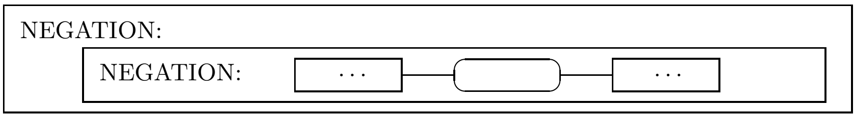
It has to be clarified *which* relations are allowed to be attached to contexts. For example, it has to be clarified whether the following graphs are conceptual graphs:





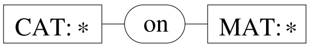
So if negation is introduced as a relation, we see the following: If arbitrary relations may be attached to concept boxes, then it is difficult to define a meaningful semantics. Thus, it has to be specified which types and relations may be linked. Sowa uses *signatures* for relation ovals which '*represents a constraint on the types of concepts that may be linked to its arcs.*' According to his definition, the first graph is not a well-formed conceptual graph. But it seems that the second graph fulfills his requirements and therefore is a well-formed conceptual graph. As an application of the relation 'neg' to Yoyo makes no sense, the meaning of this graph is not clear. Thus, if negation is introduced as a relation which may only be attached to specific concept boxes, it is difficult to define which graphs are valid conceptual graphs and which graphs are not.

The next problems arise when we ask how negation is handled in calculi or in translations to other formal languages (like Φ). If negation is expressed by special context boxes only, any calculus and any translation has to treat these special context boxes different from all other context boxes. For example, if negation is expressed by context boxes of type `Negation`, a calculus should allow the nested boxes in the following graph to be erased (and vice versa, to be introduced again):

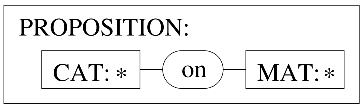


This seems to be impossible for any calculus which does not treat the negation boxes separately (like the calculus of Prediger ([45]) or any calculus which is based on projections).

If negation is expressed by contexts of type `Proposition` which are linked to a unary relation `neg`, another difficulty appears. This shall be shown by the following two conceptual graphs:



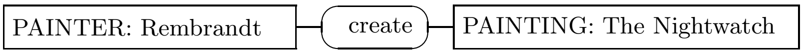
and



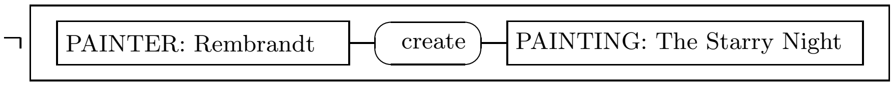
The first graph is well known: Its meaning is 'a cat is on a mat'. In particular, the graph claims to be true. The meaning of the second graph is, strictly speaking, 'there exists a proposition which states that a cat is on a mat'. Quoting a proposition changes its character. It is important to distinguish between asserting and quoting a proposition (as between using and mentioning linguistic items). Hence the meaning of the second graph is different to the meaning of the first graph. Indeed: In none of the common calculi, one graph can be derived from the other one. Hence it causes problems to express the negation of the first graph by the second graph with a relation `neg`.

Therefore, negation contexts have to be treated differently to other contexts and such an approach will lead at least to a technical overhead both in the calculus and in any translation of conceptual graphs to other formal languages.

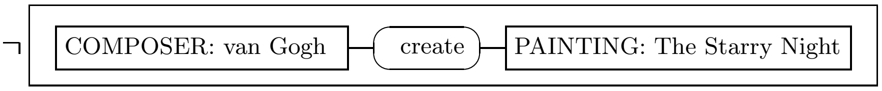
Another crucial point is that negation contexts always contain closed subgraphs (the only exception in conceptual graphs is that coreference links are allowed to cross negation contexts). This may lead to some difficulties in the semantics, i.e., the meaning of negation. In order to clarify this, we will discuss an example. Consider the following valid proposition: The painter Rembrandt created the painting The Nightwatch. This proposition can be translated to the following conceptual graph:



This graph represents not only the information that Rembrandt created The Nightwatch, but also that Rembrandt is a painter and The Nightwatch is a painting. Now, consider the painting The Starry Night instead of The Nightwatch. This painting was created by van Gogh, so the proposition 'the painter Rembrandt did not create the painting The Starry Night' is true. How can this proposition be transformed to a conceptual graph? The following graph is a first attempt:



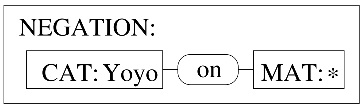
This graph is not the translation of the former proposition: In the proposition, only the verb 'to create' is negated, but in the graph, the negation box also encloses the information that Rembrandt is a painter and The Starry Night is a painting. The information in the concept boxes can also be false, as can be seen in the following graph:



This graph is true although van Gogh did create The Starry Night. In particular, this graph should not be read as

> The composer Van Gogh *did not* create the painting The Starry Night.

But this understanding is suggested when Sowa in [62] or [65] says that the meaning of `[Negation: [Cat: Yoyo]→(On)→[Mat]]`, i.e.,



is '*the graph denies that the cat Yoyo is on a mat*'. Now, the goal is to negate only the verb 'to create' in the false proposition 'the painter Rembrandt created the painting The Starry Night'. This problem has already been addressed, one approach for its solution is the following graph:

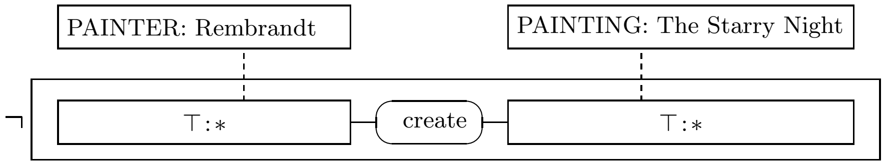


Fig. 14.1. 'The painter Rembrandt did not create the painting The Starry Night'

Indeed, this graph expresses the proposition 'the painter Rembrandt did not create the painting The Starry Night'. But obviously, the aim of making conceptual graphs easily readable and intuitively understandable is not fulfilled.

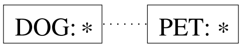
To summarize: We have both syntactical and semantical objections to introduce negation as a special context box. The main reason for this is that negation is in conceptual graphs handled as a *metalevel* operation, i.e., an operation which applies not to conceptual graphs directly, but to propositions about concept graphs (which are formalized as conceptual graphs, too). But in our view, negation is a *logical* operator and has therefore to be treated separately. As Peirce said: '*That a proposition is false is a* logical *statement about it, and therefore in a logical system deserves special treatment.*' ([41]). This yields the following conclusion: For the mathematical treatment of negation in concept graphs, in the definition of their well-formed formulas there should be an *independent* syntactical element which is used to express negation. For this reason we added the cuts of existential graphs as new syntactical element to concept graphs.

14.2 Identity Links

The next task is to express identity in concept graphs. In this section we will provide some arguments why identity is implemented in concept graphs with identity links. In order to do this, we will discuss the two main approaches how identity is to be expressed in conceptual graphs:

1. The approach of Sowa who uses coreference links or coreference sets to express identity. This approach is adopted by nearly all authors who deal with conceptual graphs.
2. The approach of Prediger who uses an equivalence relation on the generic nodes to express identity.

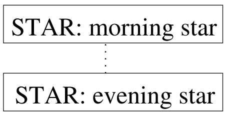
Both approaches use dotted lines for representing coreference links resp. the equivalence relation. For example, in both approaches the intuitive meaning of the graph

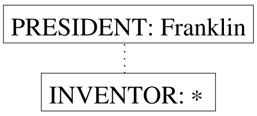


is: There is a dog which is a pet.

There must be a syntactical element in our language which is used to express identity. Before we start a detailed discussion of the approaches of Sowa and Prediger, we want to first work out some of the properties any syntactical element, which is used to express identity, should have.

The first observation is simple: Identity can not only occur between generic vertices, but it can also occur between non-generic vertices[1] and finally between a generic and a non-generic node. The best known example for the first case is Frege's example that the names 'morning star' and 'evening star' refer to the same physical object. The second example is adopted from [4]. So the following two graphs should be well-formed valid conceptual graphs:





[1] In computer science there are many systems which rely on the 'unique name assumption'. This assumption says that different object names always refer to different objects. But we follow the line of classical logic in which different names may refer to the same object.

The next observation is that we have to distinguish between *syntactical* and *semantical* identity. Syntactical identity takes place at a very basic level. It means that vertices which are coreferent are treated as if they were *one* name for an object which can appear several times in a graph (particularly identity takes place *before* any evaluation). This understanding has the following implications:

1. We have to assign the same object to coreferent vertices when the graph is evaluated in a model.
2. In the definition of the operator Φ, we should assign the same variable to coreferent vertices.

Semantical identity means that identity is handled like other relations (particularly identity is checked *during* evaluations). This understanding has the following implications:

1. It is possible to assign different objects to coreferent vertices, and the identity is not checked until the graph is evaluated in a model.
2. In the definition of the operator Φ, we should assign different variables to coreferent vertices which are equated in the formula.

Next we will argue that implementing identity on a purely syntactical is not sufficient. To understand the argument, let us assume that we have only an element in our language which expresses *syntactical* identity between vertices. The question arises whether it is possible to construct a graph which says that there are at least two things. This should be possible in every alphabet, in particular in $\mathcal{A} := (\emptyset, \{\top\}, \emptyset)$ (as we discuss approaches for expressing identity in concept graphs which differ from our solution which uses identity links, we do not have a relation $\doteq$ in our alphabet). A graph $\mathfrak{G}$ with the meaning 'there are at least two things' must contain at least two vertices v_1, v_2 which stand for these two things. Hence it has to fulfill the following condition: If $\mathfrak{G}$ is evaluated in a model, the evaluation yields true if and only if we assign different objects to v_1 and v_2. We define a relation θ on V such that a pair v_1, v_2 of two vertices is in relation θ if they are identified by our syntactical identity relation. We conclude that θ is an equivalence relation from the conditions a syntactical identity has to fulfill. Now we have two cases to discuss: $(v_1, v_2) \notin \theta$ and $(v_1, v_2) \in \theta$.

We start with the first case, i.e., $(v_1, v_2) \notin \theta$. In this case there is no connection – direct or indirect – between v_1 and v_2, neither with edges (as we have $\mathcal{R} = \emptyset$), nor with the element in our language which expresses identity. Hence no reasonable semantics will yield that the graphs evaluated to true in a model if and only if different objects are assigned to v_1 and v_2.

Now let us assume $(v_1, v_2) \in \theta$. Then it is not possible to assign different objects to v_1 and v_2, hence v_1 and v_2 cannot represent two different objects.

Both cases yield that in a language where identity is implemented on a purely syntactical level provides only the possibility to express syntactical identity, but it is impossible to express that two objects are *not* identical (thus, it is not possible to provide a graph with the meaning 'there are at least two things'). Hence a language like that is strictly weaker than FOL and therefore not sufficient for our purpose.

In the argumentation above, only in the discussion of $(v_1, v_2) \in \theta$ we needed that θ is derived from a syntactical identity. The remaining argumentation only needed that θ is an equivalence relation. For this reason the case $(v_1, v_2) \in \theta$. deserves a closer investigation.

In order to do this, we now assume that we implement identity on concept graphs by an equivalence relation θ on the set of vertices, and we assume that θ stands for a semantical identity between vertices which are in relation θ (this is the approach of Prediger). Again we are looking for a graph $\mathfrak{G}$ with the meaning 'there are at least two things', and again we suppose that $\mathfrak{G}$ contains two vertices v_1, v_2 which stand for these two things. If we have $(v_1, v_2) \notin \theta$, we have the same argumentation as for syntactical identity, hence it remains to consider $(v_1, v_2) \in \theta$. If v_1 and v_2 are placed in different contexts, it is not immediately clear how a semantics should evaluate that we have $(v_1, v_2) \in \theta$. But if v_1 and v_2 are placed in the same context, any reasonable semantics should check – when the context of v_1 and v_2 is evaluated – whether v_1 and v_2 refer to the same object. On the other hand, the proposition 'v_1 and v_2 stand for two different objects' is symmetric with regard to v_1 and v_2. Hence any reasonable syntax and semantics for concept graphs should allow to construct $\mathfrak{G}$ so that it is symmetric with regard to v_1 and v_2. Especially v_1 and v_2 are placed in the same context. But as the identity is checked in this context, too, we encode in this context (at least) the information 'there are two things which are identical' and not 'there are two things which are *not* identical', hence $\mathfrak{G}$ does not have the desired meaning. So we conclude that implementing identity by an equivalence relation is also not sufficient for our purpose.

To summarize the discussion to this point, we can draw three conclusions:

1. Identity should not be implemented only on the generic nodes.
2. Identity should not be implemented in a purely syntactical way.
3. Identity should not be implemented by an equivalence relation on the concept boxes.

In the light of these conclusions, we can now discuss the approaches of Prediger and Sowa.

Prediger presented in [44] a mathematical foundation for concept graphs (simple as well as nested). She implemented a syntactical form of identity by an equivalence relation on the generic vertices. In her approach, she did not make any arrangements for expressing negation. Therefore, it is only possible to encode *positive* information in her graphs, thus in her implementation none of the problems we discussed above arise. However, as concept graphs with cuts provide the possibility to express negations, the discussion we have carried out so far yields that we should not adopt Prediger implementation of identity for concept graphs with cuts.

The discussion of the approach of Sowa will be more extensive. He uses coreference links to express the identity of two entities. In [60] he says: '*Two concepts that refer to the same individual are coreferent. [...] To show that they are coreferent, they are connected with a dotted line, called a* coreference link.' Our discussion of Sowa's approach will be based on four examples which we provide in Fig. 14.2. The graphs $\mathfrak{G}_1$, $\mathfrak{G}_2$ and $\mathfrak{G}_3$ are taken from [60], the graph $\mathfrak{G}_4$ is taken from [62].

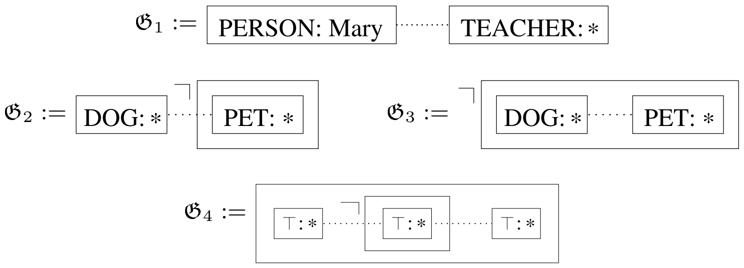


Fig. 14.2. 'Four examples of Sowa for coreference-links.

First we want to show that Sowa's approach fulfills the three conclusions we have stated at the beginning of this section. The first graph $\mathfrak{G}_1$ is an example with a coreference link between a generic and a non-generic node, i.e., we see that coreference links meet the first of our three conclusions (identity should be implemented only on the generic nodes).

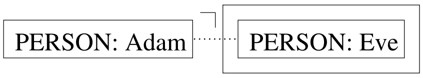
Next we have to find out whether coreference links should be understood syntactically or semantically. In [60], Sowa maps the graphs $\mathfrak{G}_2$ and $\mathfrak{G}_3$ to the formulas $\exists x.(DOG(x) \wedge \neg PET(x))$ and $\neg\exists y.(DOG(y) \wedge PET(y))$, respectively. This translation suggests that coreference links should be understood as a syntactical implementation of identity. But on the other hand, he assigns to $\mathfrak{G}_1$ the formula $\exists x.(PERSON(Mary) \wedge TEACHER(x) \wedge Mary = x)$, which suggests a semantical understanding of coreference links. This becomes more clear in [62], where Sowa provides the graph $\mathfrak{G}_4$ which is Sowa's example of a graph with the meaning 'there are at least two things'. In [62], he he translates $\mathfrak{G}_4$ to the formula $\exists x.\exists y.(\top(x) \wedge \top(y) \wedge \neg\exists z.(\top(z) \wedge x = z \wedge z = y))$. In particular, different generic vertices map to different variables, and a coreference link maps to an equation which equals the corresponding variables.

Therefore we conclude that coreference links should be understood semantically. The translations of $\mathfrak{G}_2$ and $\mathfrak{G}_3$ that Sowa provides may be abbreviations for the (semantical) translations $\exists x.(DOG(x) \wedge \neg\exists z.(PET(z) \wedge x = z))$ and $\neg\exists y.\exists z.(DOG(y) \wedge PET(z) \wedge y = z)$.

Finally, according to Sowa's declaration of coreference links, these links always connect two concept boxes. So we conclude that coreference links should not be understood as a visualisation of an equivalence relation. This can be seen even better on $\mathfrak{G}_4$. In $\mathfrak{G}_4$, we have two coreference links which are mapped to the subformulas $x = z$ and $z = y$. If these coreference links were based on an equivalence relation, we would have an additional subformula $x = y$ in our translation. Thus, $\mathfrak{G}_4$ would be mapped to the contradictory formula $\exists x.\exists y.(\top(x) \wedge \top(y) \wedge x = y \wedge \neg\exists z.(\top(z) \wedge x = z \wedge z = y))$. So $\mathfrak{G}$ would be contradictory itself and would not have the meaning Sowa intends.

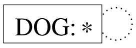
We have seen that coreference links fulfill the three requirements we have stated at the beginning of this section. Nevertheless there are some objections against coreference links which shall be explained next.

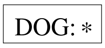
Remember that $\exists x.\exists y.(\top(x) \wedge \top(y) \wedge \neg\exists z.(\top(z) \wedge x = z \wedge z = y))$ is the translation of $\mathfrak{G}_4$. It is crucial that the equating of the variables (i.e., the subformulas $x = z$ and $z = y$) is placed *inside* the negated part of the formula. But as the link in the graph looks symmetric, it is not clear to the reader why the equating in the formula is placed inside and not outside of the negated subformula. This ambiguity can be seen even better in the following example:



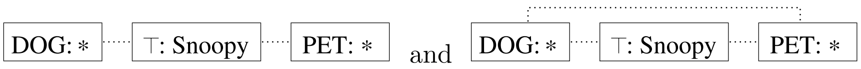
If $PERSON(Adam) \wedge \neg(Adam = Eva \wedge PERSON(Eva))$ is the translation of the graph, the resulting formula is true, but if Φ translates this graph to $PERSON(Adam) \wedge Adam = Eva \wedge \neg(PERSON(Eva))$, the resulting formula is not true. Hence, in order to understand the correct meaning of this graph, the reader must keep the implicit agreement that equality is always placed in the inner context in mind. Without this knowledge, coreference links may be misunderstood. This yields that the meaning and handling of coreference links is not intuitively clear.

The next objection we have is that the special properties of identity and hence the special properties of coreference links have to be reflected by syntax and semantics for conceptual graphs. And any calculus should contain rules which encompass the properties of identity. For example, it should be made clear whether the first of the following graphs is a valid conceptual graph. If it is a valid conceptual graph, it should be provably equivalent to the second graph:

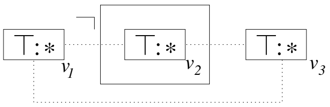




As identity is a transitive relation, the next two graphs should be provably equivalent, too:



But of course coreference links are not transitive, if some of them cross context boxes. To see this, consider the graph

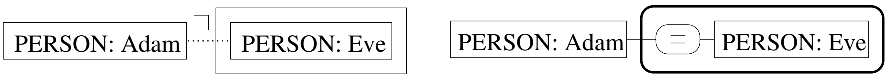


This graph is contradictory. In particular it cannot be derived from $\mathfrak{G}_4$ of Fig. 14.2. On the other hand, any calculus should allow to derive arbitrary graphs from this graph. Most calculi (e.g. the calculus Sowa provides in [59]) lack rules which allow to treat coreference links in a sound and complete way.

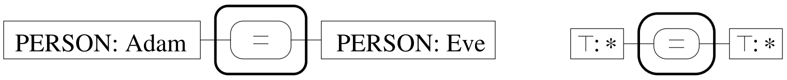
To summarize again: Introducing coreference links to express identity may lead to misunderstandings of their meaning and to gaps in their syntactical implementation, in particular in the calculus.

For these reasons, identity is introduced by a relation name $\doteq \in \mathcal{R}_2$ in concept graphs with cuts. Hence instead of using coreference links, identity is expressed – like any other relation – by edges. The edges which are labelled with the relation name $\doteq$ are therefore called *identity-links* (see Def. 2.10). This approach has the following advantages:

First, identity links can be understood as all other edges, i.e., we need no implicit agreements how identity links have to be read. Hence identity links are better understandable than coreference links. For example, compare the following conceptual graph and concept graph with the same meaning:



As we are allowed to draw cuts around edges, and as we only want to deny that Adam and Eve are identical, we can provide an even better concept graph. The left of the next two concept graphs with cuts has the meaning ‘Adam and Eve are different persons’. The other graph is a concept graph with cuts with the meaning ‘there are at least two things’. Thus, it has the same meaning as the corresponding conceptual graph of Fig. 14.2, but is much simpler.



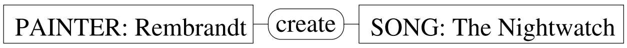
As identity has special properties, we have special rules in our calculus which encompass these properties for identity links. But as identity links are edges, in all other rules identity links are handled like all other edges. For example, the rules specialization and generalization apply to identity links, too. This is another advantage of identity links compared to coreference links.

14.3 Dominating Nodes

In this treatise we have only considered concept graphs with dominating nodes. The requirement that each concept graph has dominating nodes can be dropped, but considering concept graphs with some non-dominating nodes (i.e. nodes $v \in V$ such that there is an edge $e \in E_v$ with $ctx(e) \nleq ctx(v)$) yields immense problems in the semantics as well as in the calculus. This shall be explained in this section.

Concept graphs with non-dominating nodes can be best understood when they are mapped to FOL. We start with the following (false) graph:

$\mathfrak{G}_1 :=$

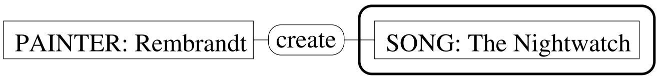


We have

$$\Phi(\mathfrak{G}_1) = PAINTER(R) \wedge create(R, nw) \wedge SONG(nw)$$

(in the formulas, we abbreviate *Rembrandt* by R and $The\ Nightwatch$ by nw to get shorter formulas). It is true that Rembrandt created The Nightwatch, but The Nightwatch is of course the famous painting, not a song. Hence $\mathfrak{G}_1$ is false. The wrong information that The Nightwatch is a song is encoded by the concept box $\boxed{\text{SONG: The Nightwatch}}$, so this box should be negated. This would yield the following concept graph with a non-dominating node:

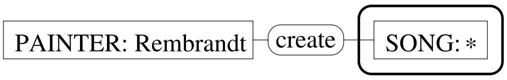
$\mathfrak{G}_2 :=$



If the definition of Φ is expanded to concept graphs without dominating nodes, it should yield:

$$\Phi(\mathfrak{G}_2) = PAINTER(R) \wedge create(R, nw) \wedge \neg SONG(nw) \quad .$$

This translation is straight forward. In fact, we can improve the readability of some graphs if we allow concept graphs to have non-dominating nodes, too. But as soon as generic markers are involved, the situation is completely different. To see this, we replace the name 'The Nightwatch' by the generic marker '$*$', i.e., we consider

$\mathfrak{G}_3 :=$ 

If we simply apply the operator $\mathit{\Phi}$ to $\mathfrak{G}_3$, we get the formula

$$\mathit{\Phi}(\mathfrak{G}_3) = PAINTER(R) \wedge create(R,x) \wedge \neg\exists x.SONG(x) \quad .$$

The first occurence of the variable x is not in the scope of the quantifiers $\exists x$, i.e., it appears free in the formula. It is clear that the formula above cannot be a reasonable translation of $\mathfrak{G}_3$ for two reasons:

- The different occurences of x may refer to different objects.
- Concept graphs should always be translated to *closed* formulas.

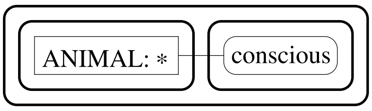
Thus the definition of $\mathit{\Phi}$ has to be modified when we consider concept graphs without dominating nodes, too.

To get a reasonable definition of $\mathit{\Phi}$, the intended meaning of $\mathfrak{G}_3$ has to be made clear. A first idea may be the following: 'The painter Rembrandt did not create a song'. But this statement is true even if Rembrandt did not create *anything*. But in $\mathfrak{G}_3$, the relation oval –(create)– appears directly on the sheet of assertion, hence the graph postulates a relationship of type 'create' between Rembrandt and another object. This object is quantified by a generic marker. As the first attempt to translate $\mathfrak{G}_3$ shows: This quantification does not take place *inside* the cut. Thus it takes place *outside* the cut, i.e., on the sheet of assertion. The type of the object is described by the concept box $\boxed{\text{SONG: } *}$ which is placed in the cut. From the discussion we conclude that the type of the concept box is negated, but the quantification is not. So the right understanding of $\mathfrak{G}_3$ is the following: The painter Rembrandt created something which is not a song. Hence the right translation to a FOL-formula should be:

$$\mathit{\Phi}(\mathfrak{G}_3) := \exists x.(PAINTER(R) \wedge create(R,x) \wedge \neg SONG(x))$$

The reason for this translation is the following: Although the generic marker appears in a cut, the fact that the concept box $\boxed{\text{SONG: } *}$ is linked to the relation oval –(create)– which is placed on the sheet of assertion yields that *the scope* of the generic marker goes beyond the cut to the sheet of assertion, too.

This consideration shall be exemplified with another graph. We consider

$\mathfrak{G}_4 :=$ 

Again the question arises where the existential quantification of the generic marker takes place, i.e., where the scope of the generic marker is. For the same reason as in the last example, it cannot be the (inner) cut where the box $\boxed{\text{ANIMAL: } *}$ is placed. An analogous argument yields that it cannot be the cut where the oval —(conscious) is placed, either. So it must be the least context which encloses the concept box as well as the attached relation oval. This is the outermost cut, thus $\mathfrak{G}_4$ should be translated as follows:

$$\varPhi(\mathfrak{G}_4) := \neg\exists x.(\neg ANIMAL(x) \wedge \neg conscious(x))$$

The discussion so far yields that the understanding of concept graphs without dominating nodes rapidly becomes difficult and complicated, but we can at least provide a reasonable extension of the definition of $\varPhi$ to concept graphs without dominating nodes where we take the scope of generic concept boxes into account.

Extended definition of $\varPhi$.

For each generic node $v \in V$ (i.e., $\rho(v) = *$) we set

$$\underline{scope(v)} := \min\{c \in Cut \cup \{\top\} \mid \{v\} \cup V_e \subseteq \leq[c]\}$$

($scope(v)$ is the context in which v is quantified).

Let $\mathfrak{G} := (V, E, \nu, \top, Cut, area, \kappa, \rho)$ be a simple concept graph with cuts and variables. We set $Free(\mathfrak{G}) := \{\alpha \in \mathrm{Var} \mid \exists v \in V.\, \rho(v) = *_\alpha\}$, and we assign to each vertex $v \in V^*$ a fresh variable $\alpha_v \notin Free(\mathfrak{G})$, so that we can define the following mapping $\varPhi_t$ on V:

$$\varPhi_t(v) := \begin{cases} \alpha_v \text{ for } \rho(v) = * \\ \alpha \text{ for } \rho(v) = *_\alpha \text{ and } \alpha \in \mathrm{Var} \\ g \text{ for } \rho(v) = g \text{ and } g \in \mathcal{G} \end{cases}$$

We need a further variable $\alpha_{empty} \notin Free(\mathfrak{G}) \cup \{\alpha_v \mid \rho(v) = *\}$. Now, inductively over the tree $Cut \cup \{\top\}$, we assign to each cut $c \in Cut \cup \{\top\}$ a formula $\varPhi(\mathfrak{G}, c)$. So let c be a context such that $\varPhi(\mathfrak{G}, d)$ is already defined for each cut $d < c$. First, we define a formula f which encodes all edges and vertices which are directly enclosed by c. Hence, if c does not directly enclose any edges or vertices, simply set $f := (\exists \alpha_{empty}.\top(\alpha_{empty}))$. Otherwise, let f be the conjunction of the following atomic formulas:

$$\kappa(w)\underline{(}\varPhi_t(w)\underline{)} \text{ with } w \in V \cap area(c),$$

$$\varPhi_t(w_1)\underline{=}\varPhi_t(w_2) \text{ with } k \in E^{id} \cap area(c) \text{ und } \nu(k) = (w_1, w_2), \quad \text{and}$$

$$\kappa(e)\underline{(}\varPhi_t(w_1)\underline{,}\ldots\underline{,}\varPhi_t(w_j)\underline{)} \text{ with } e \in E^{nonid} \cap area(c) \text{ and } \nu(e) = (w_1, \ldots, w_j).$$

Let $v_1, \ldots, v_n$ be the vertices of $\mathfrak{G}$ with $scope(v_i) = c$ (this is the main difference to the Definition of $\varPhi$ in Sect. 9.2). Let $area(c) \cap Cut = \{c_1, \ldots, c_l\}$

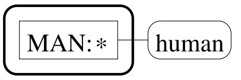
(by induction, we already assigned formulas to these cuts). If $l = 0$, set $\Phi(\mathfrak{G}, c) := \underline{\exists}\alpha_{v_1}\underline{.}\ldots\underline{\exists}\alpha_{v_n}\underline{.}f$, otherwise set

$$\Phi(\mathfrak{G}, c) := \underline{\exists}\alpha_{v_1}\underline{.}\ldots\underline{\exists}\alpha_{v_n}\underline{.}\underline{(}f\underline{\wedge}\underline{\neg}\Phi(\mathfrak{G}, c_1)\underline{\wedge}\ldots\underline{\wedge}\underline{\neg}\Phi(\mathfrak{G}, c_l)\underline{)} \quad ,$$

Finally set $\Phi(\mathfrak{G}) := \Phi(\mathfrak{G}, \top)$, and the definition of Φ is finished.

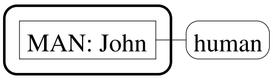
We have seen that in the definition of Φ, we have to respect the scope of generic nodes. The same holds for the semantics, too. This can easily be seen with $\mathfrak{G}_3$. If we want to apply the endoporeutic method to $\mathfrak{G}_3$, we start its evaluation on the sheet of assertion. But we can only evaluate the sheet of assertion, when we have already assigned an object to the generic marker of the vertex *which is placed in the cut.* This is – according to the definition of $\models_{endo}$ in Def. 9.3 – not the case when the sheet of assertion is evaluated. So we see that the endoporeutic method only works properly for graphs with dominating nodes. Similar to the definition of Φ, it would be possible to extend the definition of $\models_{endo}$ to concept graphs without dominating nodes, but this would result in a very technical, complicated and, above all, unintuitive semantics. This will not be done here. In the rest of this section, we will use the extended definition of Φ to describe the meaning of concept graphs without domination nodes.

Finally, we investigate whether the rules of the calculus can be applied to concept graphs without dominating nodes. For this we analyse some 'naive' applications of some rules of the calculus. We start with the following graph:



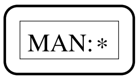
The translation of this graph is $\exists x.(human(x) \wedge \neg(MAN(x)))$, i.e., the meaning of this graph is 'there exists an object which is human, but which is not a man'. As women exist, this graph is true.

The concept box $\boxed{\text{MAN: } *}$ is placed in a negative cut, hence a naive application of the rule of specialization allows us to replace the generic marker by the object name 'John'. This yields the following graph:



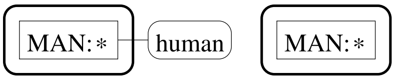
This graph translates to $human(John) \wedge \neg MAN(John)$, hence it is *not* true. So we see that the specialization-rule is, when applied to concept graph without dominating nodes, not sound any longer. An analogous example would yield the same result for the generalization-rule.

As the relation oval is placed on the sheet of assertion, we can erase it with a naive application of the erasure-rule. This yields



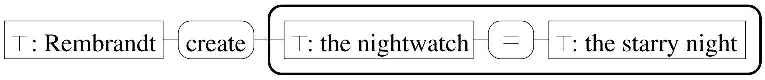
Again we get an false graph, hence the erasure-rule (and the insertion-rule as well) is not sound any longer, either.

The same holds for the iteration-rule. A naive application of the iteration-rule allows us to draw a copy of the cut and the enclosed concept box on the sheet of assertion. This yields

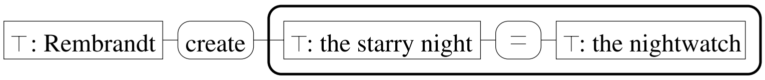


As this is again a false graph, we conclude that the iteration-rule (and the deiteration-rule as well) is not sound any longer, either.

A closer observation of the examples above yields that they all fail for the same reason: By the use of one of the rules, the scope of the generic marker $*$ is changed. But even in Alpha, where we do not have any generic markers, we run into problems. To see this, consider

$\mathfrak{G}_a :=$ 

An application of the Alpha-rule ‘exchanging references’ yields:

$\mathfrak{G}_b :=$ 

Note that $\mathfrak{G}_a$ and $\mathfrak{G}_b$ are isomorphic except for the their contexts. Furthermore, in both graphs we find that the subgraph which is enclosed by the cut evaluates to be false (hence in both graphs, the cut evaluates to be true). Nevertheless we have that $\mathfrak{G}_a$ is true, but $\mathfrak{G}_b$ is false! So even if we demand that the rules do not change the scope of any generic node, they may fail. Furthermore we see in this example that the idea of an ‘isomorphism except a context’ does not work properly any longer if it is applied to concept graphs without dominating nodes. In particular, nearly none of the soundness-proofs, which rely heavily on isomorphisms except a context, can be applied to concept graphs without dominating nodes.

To summarize: If we consider concept graphs without dominating nodes, nearly every notion, definition and theorem in this treatise has to be amended and modified. It is likely that the semantics and the calculus will become much more technical and they loose much of their intuitive comprehensibility and understandability. On the other hand, concept graphs with dominating nodes and concept graphs without dominating nodes have the same expressive power, so we do not achieve higher expressiveness if we consider concept graphs some non-dominating nodes. For this reason this treatise is restricted to concept graphs with dominating nodes.

14.4 Peirce-Style Calculi

The calculi which are presented in this treatise are based on the calculi of Peirce for the parts Alpha and Beta of existential graphs. The first five rules are a direct translation of Peirce's rules for concept graph with cuts, but it was necessary to add further rules. These additional rules are designed such that the whole calculus is still a 'Peirce-style calculus'. This shall be explained in this section.

The rules of Peirce are rather different to the kind of rules which are used in FOL. A first reason for this is the shape and construction of existential graphs (and hence for concept graphs with cuts). We have the following main differences between existential graphs and FOL-systems:

- FOL-systems use different variables, i.e., different symbols, to speak about indefinite objects and to range over our universe of discourse. In existential graphs, only one symbol, the line of identity, is used instead.
- FOL-systems are usually inductively defined (see for example the definition of FOL-formulas in Chap. 8), existential graphs are not.
- As FOL-systems are usually inductively defined, the concept of free variables is needed. There is no counterpart of free variables in existential graphs

The use of free variables and the use of different variable names have to be captured in FOL-systems by rules. There are no corresponding rules for existential graphs.

The inductive construction of FOL-formulas causes a main property of common FOL-calculi: The usual rules of FOL-calculi, even the natural deduction rules of Gentzen, only allow transformations at the outermost parts of a formula (i.e., they do not allow to modify deeper nested subformulas). In contrast, the rules of Peirce allow transformations in arbitrary deeply nested cuts. For this reason the rules of Peirce are very powerful (and it is not trivial to comprehend and prove their soundness).

The next observation is that the rules of Peirce are grouped in pairs. There are two kinds of rules: First, we have rules which do not change the informational content of a graph (Sowa calls rules like these *equivalence rules*). Second, we have rules which (possibly) weaken the informational content of a graph

(Sowa calls rules like these *generalization rules*[2]). Both kinds of rules are grouped in pairs:

- **equivalence rules:** Each equivalence rule may be carried out in both directions and in arbitrary contexts. To see an example for this, consider the pair iteration/deiteration. The deiteration-rule is simply the inverse direction of the iteration-rule and vice versa. So they represent one rule which can be carried out in both directions. Furthermore, they may be executed in arbitrary contexts.

  The same holds for the double cut rule. In fact this rule can be seen as a pair of rules, too: Inserting a double cut and erasing a double cut. Again each of these two rules is simply the inverse direction of the other rule. And again, the double cut rule may be executed in arbitrary contexts.
- **generalization rules:** Generalization rules are also grouped in pairs such that each of these two rules is simply the inverse direction of the other rule. The crucial difference to equivalence rules is that they may not be carried out in arbitrary contexts. One of the rules may only be executed in *positive* contexts, and the other rule may only be executed in *negative* contexts.

  The only example for this in the Peirce-calculus for existential graphs is the pair erasure/insertion. For concept graphs with cuts, the pair specialization/generalization is another example.

Finally, we find another main difference to common FOL-calculi. First order logic is built upon propositional logic. The step from a calculus for propositional logic to a calculus for first order logic is taken by *adding* rules (which capture the properties of variables, identity, quantification, etc). In the system of existential graphs, in the step taken from Alpha to Beta (which corresponds to the step from propositional logic to first order logic), *no* rules are added to the Alpha-calculus, but the rules of the Alpha-calculus are *refined* in order to get a set of rules for Beta.

To summarize:

- The calculi of Peirce are grouped in pairs of rules which are inverse to each other and which may be executed in arbitrary contexts (for equivalence rules) resp. in arbitrary positive or arbitrary negative contexts (for generalization rules).
- The Beta-calculus is not derived from the Alpha-calculus by adding new rules, but by refining the existing rules.

These two properties cause the elegance of Peirce's calculi. For this reason we decided to design the calculi for concept graphs with cuts such that it

[2] Please do not confuse the classification of generalization rules with the generalization rule of the concept graph calculus which allows to generalize the concept name or the referent of a positive nested concept box.

fulfill these properties, too. Both calculi (for non-existential and existential concept graphs with cuts) contain the same set of rules. This set of rules is grouped in pairs, as it is shown in Fig. 14.3.

Pair of rules	equivalence or generalization
erasure and insertion	generalization
iteration and deiteration	equivalence
double cut	equivalence
generalization and specialization	generalization
exchanging references (self-symmetric)	equivalence
$\top$-erasure and $\top$-insertion	equivalence
identity-erasure and identity-insertion	equivalence
merging two vertices and splitting a vertex	equivalence

Fig. 14.3. The rules for existential concept graphs with cuts.

It should be remarked that this is a *redundant* set of rules. First, for most pairs of rules it would be sufficient to add only one of the two rules to the calculus (see Proposition 6.20). But we added both rules to the calculus to keep the calculus symmetric.

Second, there is no need that all rules can be carried out in arbitrary (or arbitrary positive or arbitrary negative) contexts. For example, it would be sufficient to formulate the rules $\top$-erasure and $\top$-insertion not for arbitrary contexts, but only for the sheet of assertion (i.e., for each object name g, an isolated vertex $\boxed{\top : g}$ may be erased from or inserted to the sheet of assertion). But in order to get a Peirce-style calculus, they are still formulated for arbitrary contexts.

References

1. F. Baader, R. Molitor, S. Tobies: *The Guarded Fragment of Conceptual Graphs.* RWTH LTCS-Report.
`http://www-lti.informatik.rwth-aachen.de/Forschung/Papers.html`
See also [2].
2. F. Baader, R. Molitor, S. Tobies: *Tractable and Decidable Fragments of Conceptual Graphs.* In: W. Tepfenhart, W. Cyre (Eds.): Conceptual Structures: Standards and Practices. LNAI 1640, Springer Verlag, Berlin–New York 1999, 480–493.
Excerpt of [1].
3. J. Barwise, J. Perry: *Situations and Attitudes.* MIT Press, Cambridge 1983 (deutsch: Situationen und Einstellungen: Grundlagen der Situationssemantik, De Gruyter, Berlin 1987).
4. R. B. Brandom: *Begründen und Begreifen. Eine Einführung in den Inferentialismus.* Suhkamp, 2001.
5. W. Brugger: *Philosophisches Wörterbuch.* Herder Verlag, Freiburg, 14. Aufl. 1976.
6. R. W. Burch: *A Peircean Reduction Thesis: The Foundations of Topological Logic.* Texas Tech University Press, 1991.
7. M. Chein, M.-L. Mugnier: *Conceptual Graphs: Fundamental Notions.* Revue d'Intelligence Artificielle 6, 1992, 365–406.
8. M. Chein, M.-L. Mugnier: *Conceptual Graphs are also Graphs.* Rapport de Recherche 95003, LIRMM, Université Montpellier II, 1995.
9. M. Chein, M.-L. Mugnier: *Representer des Connaissances et Raisonner avec des Graphes.* Revue d'intelligence Artificielle 10, 1996, 7–56.
10. M. Chein, M.-L. Mugnier: *Positive Nested Conceptual Graphs.* In: D. Lukose et al. (Eds.): Conceptual Structures: Fulfilling Peirce's Dream. LNAI 1257, Springer Verlag, Berlin–New York 1997, 95–109.
11. M. Chein, M.-L. Mugnier,G. Simonet: *Nested Graphs: A Graph-based Knowledge Representation Model with FOL Semantics.* Rapport de Recherche, LIRMM, Université Montpellier II, 1998.
12. D. van Dalen: *Logic and Structure.* Springer Verlag, Berlin – Heidelberg 1996.
13. F. Dau: *Negations in Simple Concept Graphs.* In: B. Ganter, G. W. Mineau (Eds.): Conceptual Structures: Logical, Linguistic, and Computational Issues. LNAI 1867, Springer Verlag, Berlin–New York 2000, 263–276.
14. F. Dau: *Concept Graphs and Predicate Logic.* In: H. S. Delugach, G. Stumme (Eds.): Conceptual Structures: Broadening the Base. LNAI 2120, Springer Verlag, Berlin–New York 2001, 72–86.

15. F. Dau: *An Embedding of Existential Graphs into Concept Graphs with Negations.* To appear in D. Corbett, U: Priss (Eds.): Conceptual Structures: Integration and Interfaces, LNAI, Springer Verlag, Berlin–Heidelberg 2002.
16. K. Devlin: *Logic and Information.* Cambridge University Press, 1991.
17. U. Friedrichsdorf: *Einführung in die klassische und intensionale Logik.* Vieweg, 1992.
18. B. Ganter, R. Wille: *Formale Begriffsanalyse. Mathematische Grundlagen.* Springer Verlag, Berlin–Heidelberg 1996.
19. B. Ganter, R. Wille: *Formal Concept Analysis: Mathematical Foundations.* Springer, Berlin–Heidelberg–New York 1999.
20. B. Ganter, R. Wille: *Contextual Attribute Logic.* In: W. Tepfenhart, W. Cyre (Eds.): Conceptual Structures: Standards and Practices. LNAI 1640, Springer Verlag, Berlin–New York 1999, 377–388.
21. B. Groh, R. Wille: *Lattices of Triadic Concept Graphs.* In: B. Ganter, G. W. Mineau (Eds.): Conceptual Structures: Logical, Linguistic, and Computational Issues. LNAI 1867, Springer Verlag, Berlin–New York 2000, 263–276.
22. E. M. Hammer: *Logic and Visual Information.* CSLI Publications, Stanford, California, 1995.
23. E. M. Hammer: *Semantics for Existential Graphs.* Journal Philosohpical Logic, Vol. 27, 1998, 489–503.
24. J. Hintikka: *The Game of Language: Studies in Game-Theoretical Semantics and its Applications.* D. Reidel, Dordrecht.
25. J. Hintikka: *The Place of C S. Peirce in the History of Logical Theory.* In: Brunning and Forster (1997), 13–33.
26. T. Jech: *Set Theory.* Academic Press 1978.
27. G. N. Kerdiles: *Saying it with Pictures: A Logical Landscape of Conceptual Graphs.* ILLC Dissertation Series, DS 2001-09.
28. J. Klinger: *Simple Semiconcept Graphs: A Boolean Logic Approach.* In: H. S. Delugach, G. Stumme (Eds.): Conceptual Structures: Broadening the Base. LNAI 2120, Springer Verlag, Berlin–Heidelberg 2001, 115-128.
29. J. Klinger: *Semiconcept Graphs: Syntax and Semantics.* Diplomarbeit, FB Mathematik, TU Darmstadt 2001.
30. J. Klinger: *Semiconcept Graphs with Variables.* To appear in D. Corbett, U: Priss (Eds.): Conceptual Structures: Integration and Interfaces, LNAI, Springer Verlag, Berlin–Heidelberg 2002.
31. L. Kreiser, S. Gottwald, W. Stelzer: *Nichtklassische Logik.* Akademie–Verlag Berlin, 1990.
32. A. Levy: *Basic Set Theory.* Springer Verlag 1979.
33. D. Lukose, R. Kremer: *Knowledge Engineering: Knowledge Representation.* `http://www.cpsc.ucalgary.ca/~kremer/courses/CG/`
34. B. Mates: *Elementare Logik.* Vandenhoeck & Ruprecht 1969.
35. J. McCarthy: *Notes on Formalizing Context.* In: Proccedings of the IJCAI 1993, 555–560.
36. E. Mendelson: *Introduction to Mathematical Logic.* D. Van Nostrand Company, Princeton 1964
37. J. D. Monk: *Introduction to Set Theory*, McGraw–Hill 1969.
38. M.-L. Mugnier: *Knowledge Representation and Reasonings Based on Graph Homomophism.* In: B. Ganter, G. W. Mineau (Eds.): Conceptual Structures: Logical, Linguistic and Computational Issues. LNAI 1867, Springer Verlag, Berlin–New York 2000, 172–192.

39. K. Oehler: *Charles Sanders Peirce.* Beck'sche Reihe, 1993.
40. H. Pape: *Charles S. Peirce: Phänomen und Logik der Zeichen.* German translation of Peirce's *Syllabus of Certain Topics of Logic.*. Suhrkamp Verlag Wissenschaft, 1983.
41. C. S. Peirce: *Reasoning and the Logic of Things. The Cambridge Conferences Lectures of 1898.* Cambridge, Massachusetts, London, England, Ed. by K. L. Ketner, H. Putnam, Harvard Univ. Press, Cambridge 1992.
42. C. S. Peirce, *MS 478.* Collected Papers, 4.394-417. Harvard University Press, Cambrigde, Massachusetts.
43. C. S. Peirce, J. F. Sowa: *Existential Graphs: MS 514 by Charles Sanders Peirce with Commentary by John F. Sowa.* `http://www.jfsowa.com/peirce/ms514.htm`
44. S. Prediger: *Kontextuelle Urteilslogik mit Begriffsgraphen. Ein Beitrag zur Restrukturierung der mathematischen Logik.* Shaker Verlag 1998.
45. S. Prediger: *Simple Concept Graphs: A Logic Approach.* In: M, -L. Mugnier, M. Chein (Eds.): Conceptual Structures: Theory, Tools and Applications. LNAI 1453, Springer Verlag, Berlin–New York 1998, 225–239.
46. S. Prediger: *Mathematische Logik in der Wissensverarbeitung: Historisch-Philosophische Gründe für eine Kontextuelle Logik.* Mathe. Semesterberichte 47, 2000.
47. S. Prediger: *Terminologische Merkmalslogik in der Formalen Begriffsanalyse.* In: G. Stumme, R. Wille (Eds.): Begriffliche Wissensverarbeitung: Methoden und Anwendungen. Springer–Verlag, Heidelberg, 2000, 99–124.
48. S. Prediger: *Nested Concept Graphs and Triadic Power Context Families: A Situation–based Contextual Approach.* In: B. Ganter, G. W. Mineau (Eds.): Conceptual Structures: Logical, Linguistic, and Computational Issues. LNAI 1867, Springer Verlag, Berlin–New York 2000, 263–276.
49. S. Prediger, R. Wille: *The Lattice of Concept Graphs of a Relationally Scaled Context.* In: W. Tepfenhart, W. Cyre (Eds.): Conceptual Structures: Standards and Practices. LNAI 1640, Springer Verlag, Berlin–New York 1999, 377–388.
50. A. Preller, M.-L. Mugnier, M. Chein: *A Logic for Nested Graphs.* Research Report 95038, LIRMM, Université Montpellier II, 1995.
51. W. Rautenberg: *Einführung in die mathematische Logik.* Vieweg Lehrbuch Mathematik, 1996.
52. D. D. Roberts: *The Existential Graphs of Charles S. Peirce.* Mouton, The Hague, Paris, 1973.
53. D. D. Roberts: *The Existential Graphs.* Computers Math. Appl.., Vol. 23, No. 6–9, 1992, 639–63.
54. J. Seligman, L. S. Moss: *Situation Theory.* In: J. van Benthem, A. ter Meulen: Handbook of Logic and Language, Elsevier Science, Amsterdam 1997.
55. S. Shin: *Reconstituting Beta Graphs into an Efficacious System.* In: Journal of Logic, Language and Information, Vol. 8, No. 3, July 1999.
56. S. J. Shin: *The Iconic Logic of Peirce's Graphs.* Bradford Book, Massachusetts, 2002.
57. J. R. Shoenfield: *Mathematical Logic.* Addison–Wesley 1967.
58. G. Simonet: *Two FOL-Semantics for Simple and Nested Conceptual Graphs.* In: M, -L. Mugnier, M. Chein (Eds.): Conceptual Structures: Theory, Tools and Applications. LNAI 1453, Springer Verlag, Berlin–New York 1998, 240–254.
59. J. F. Sowa: *Conceptual Structures: Information Processing in Mind and Machine.* Addison Wesley Publishing Company Reading, 1984.

60. J. F. Sowa: *Conceptual Graphs Summary.* In: T. E. Nagle, J. A. Nagle, L. L. Gerholz, P. W. Eklund (Eds.): Conceptual Structures: current research and practice, Ellis Horwood, 1992, 3–51.
61. J. F. Sowa: *Syntax, Semantics, and Pragmatics of Contexts.* In: G. Ellis, R. Levinson, W. Rich, J. F. Sowa (Eds.): Conceptual Structures: Applications, Implementation, and Theory. LNAI 954, Springer Verlag, Berlin, 1995, 1–15.
    See also [66]
62. J. F. Sowa: *Logic: Graphical and Algebraic.* Manuskript, Croton-on-Hudson 1997.
63. J. F. Sowa: *Peircean Foundation for a Theory of Context.* In: D. Lukose et al. (Hrsg.): Conceptual Structures: Fulfilling Peirce's Dream. LNAI 1257, Springer Verlag, Berlin–New York 1997, 41–64.
    See also [66]
64. J. F. Sowa: *Conceptual Graphs: Draft Proposed American National Standard.* In: W. Tepfenhart, W. Cyre (Eds.): Conceptual Structures: Standards and Practices. LNAI 1640, Springer Verlag, Berlin–New York 1999, 1–65.
    See also [67]
65. J. F. Sowa: *Knowledge Representation: Logical, Philosophical, and Computational Foundations.* Brooks Cole Publishing Co., Pacific Grove, CA, 2000.
66. J. F. Sowa: *Semantic Foundations of Contexts.*
    `http://www.jfsowa.com/ontology/contexts.htm`
    This paper is a revised merger of [61] and [63].
67. J. F. Sowa: *Conceptual Graphs: Draft Proposed American National Standard.*
    Old version: `http://www.jfsowa.com/cg/cgdpansw.htm`
    New version: `http://www.jfsowa.com/cg/cgstandw.htm`
    See also [64]
68. J. Tappe: *Simple Concept Graphs with Universal Quantifiers.* FB 4 Preprint, TU Darmstadt. 2000.
69. H. P. Tuschik, H. Wolter: *Mathematische Logik – kurzgefasst.* BI Wissenschaftsverlag, Mannheim–Leipzig–Wien–Zürich, 1994.
70. M. Wermelinger: *Conceptual Graphs and First-Order Logic.* In: G. Ellis, R. Levinson, W. Rich, J. F. Sowa (Eds.): Conceptual Structures: Applications, Implementation, and Theory. LNAI 954, Springer Verlag, Berlin 1995, 323–337.
71. R. Wille: *Concept Lattices and Conceptual Knowledge Systems.* In: Computers & Mathematics with Applications. 23 (1992), 493–515.
72. R. Wille: *Plädoyer für eine Philosophische Grundlegung der Begrifflichen Wissensverarbeitung.* In: R. Wille, M. Zickwolff (Eds.): Begriffliche Wissensverarbeitung: Grundfragen und Aufgaben. B.I.–Wissenschaftsverlag, Mannheim, 1994, 11–25.
73. R. Wille: *Restructuring Mathematical Logic: An Approach Based on Peirce's Pragmatism.* In: A. Ursini, P. Agliano (Eds.): Logic and Algebra. Marcel Dekker, New York 1996, 267–281.
74. R. Wille: *Conceptual Structures of Multicontexts.* In: P. W. Eklund, G. Ellis, G. Mann (Hrsg.): Conceptual Structures: Knowledge Representation as Interlingua. LNAI 115, Springer Verlag, Berlin–Heidelberg–New York 1996, 23–39.
75. R. Wille: *Conceptual Graphs and Formal Concept Analysis.* In: D. Lukose et al. (Hrsg.): Conceptual Structures: Fulfilling Peirce's Dream. LNAI 1257, Springer Verlag, Berlin–New York 1997, 290–303.

76. R. Wille: *Triadic Concept Graphs.* In: M.-L. Mugnier / M. Chein (Eds.): Conceptual Structures: Theory, Tools and Application. LNAI 1453, Springer Verlag, Berlin–Heidelberg 1998.
77. R. Wille: *Contextual Logic Summary.* In: G. Stumme (Ed.): Working with Conceptual Structures. Contributions to ICCS 2000. Shaker, Aachen 2000, 265–276.
78. R. Wille: *Boolean Concept Logic.* In: B. Ganter, G. W. Mineau (Eds.): Conceptual Structures: Logical, Linguistic, and Computational Issues. LNAI 1867, Springer Verlag, Berlin–New York 2000, 263–276.
79. R. Wille: *Boolean Judgement Logic.* In: H. S. Delugach, G. Stumme (Eds.): Conceptual Structures: Broadening the Base. LNAI 2120, Springer Verlag, Berlin–Heidelberg 2001, 115–128.
80. R. Wille: *Existential Concept Graphs of Power Context Families.* To appear in D. Corbett, U: Priss (Eds.): Conceptual Structures: Integration and Interfaces, LNAI, Springer Verlag, Berlin–Heidelberg 2002.
81. R. Wille: *Conceptual Contents as Information – Basics for Contextual Judgement Logic.* In: A. d. Moor, W. Lex, B. Ganter (Eds.): Conceptual Structures for Knowledge Creation and Communication, LNAI, Springer Verlag, Berlin–Heidelberg 2003.

Index