Lecture Notes in Artificial Intelligence 2891

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

Jaeho Lee Mike Barley (Eds.)

Intelligent Agents and Multi-Agent Systems

6th Pacific Rim International Workshop
on Multi-Agents, PRIMA 2003
Seoul, Korea, November 7-8, 2003
Proceedings

Springer

Series Editors

Jaime G. Carbonell, Carnegie Mellon University, Pittsburgh, PA, USA
Jörg Siekmann, University of Saarland, Saarbrücken, Germany

Volume Editors

Jaeho Lee
University of Seoul
Department of Electrical and Computer Engineering
90 Cheonnong-dong, Tongdaemun-gu, Seoul 130-743, Korea
E-mail: jaeho@uos.ac.kr

Mike Barley
University of Auckland
Department of Computer Science
Private Bag 92019, Auckland, New Zealand
E-mail: barley@cs.auckland.ac.nz

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.2.11, I.2, C.2.4, D.2, F.3

ISSN 0302-9743
ISBN 3-540-20460-1 Springer-Verlag Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by PTP-Berlin, Protago-TeX-Production GmbH
Printed on acid-free paper SPIN: 10969090 06/3142 5 4 3 2 1 0

Preface

Five years ago, with excitement and uncertainty, we witnessed the birth of PRIMA (Pacific Rim International Workshop on Multi-Agents). The first PRIMA in 1998 has now grown into PRIMA 2003, the 6th Pacific Rim International Workshop on Multi-Agents in Seoul, Korea. During a period of five years, the notion of agent research has grown so much that we hear the term *agent* on a daily basis. Various fields such as business, the Web, software engineering, on-line games and such are now using the term *agent* as a placeholder, just like the term *object* is used in the object-oriented paradigm. On the other hand, the research area has extended toward real applications, such as the Semantic Web and ubiquitous computing. The themes of PRIMA 2003 reflected the following trends:

- agent-based electronic commerce, auctions and markets
- agent architectures and their applications
- agent communication languages, dialog and interaction protocols
- agent ontologies
- agent programming languages, frameworks and toolkits
- agentcities
- agents and grid computing
- agents and peer computing
- agents and the Semantic Web
- agents and Web services
- artificial social systems
- conflict resolution and negotiation
- evaluation of multi-agent systems
- languages and techniques for describing (multi-)agent systems
- meta modeling and meta reasoning
- multi-agent planning and learning
- multi-agent systems and their applications
- social reasoning, agent modeling, and organization
- standards for agents and multi-agent systems
- teams and coalitions
- ubiquitous agents

PRIMA 2003 in Seoul, Korea hoped to build on the wonderful success of its predecessors, PRIMA 1998 in Singapore, PRIMA 1999 in Kyoto, Japan, PRIMA 2000 in Melbourne, Australia, PRIMA 2001 in Taipei, Taiwan, and PRIMA 2002 in Tokyo, Japan. We received 44 submissions to be considered. Each paper was carefully reviewed by three internationally renowned agent researchers chosen from the program committee (PC) members. There were 18 papers selected to be printed in this volume. We would like to thank all the authors for submitting their papers to PRIMA 2003 and the PC members for

their diligent efforts to review those papers. Also, we are so grateful to have had our invited speakers, Ronald Arkin (Georgia Institute of Technology), Edmund Durfee (University of Michigan), Nick Gibbins (University of Southampton), and Hiroshi Ishiguro (Osaka University). Last, but not least, we would like to express our thanks to the editorial staff of Springer-Verlag for publishing this volume in the Lecture Notes in Computer Science series and the Institute of Information Technology Assessment (IITA), Korea for their generous financial support for this workshop.

November 2003

Jaeho Lee
Mike Barley

Organization

PRIMA 2003 was organized by the Korea Intelligent Information Systems Society (KIISS) and the Electronics and Telecommunications Research Institute (ETRI).

General Chair

Jin Hyung Kim
Computer Science Department
KAIST (Korea Advanced Institute of Science and Technology), Korea
jkim@cs.kaist.ac.kr

Program Co-chairs

Jaeho Lee
Department of Electrical and Computer Engineering
University of Seoul
90 Cheonnong-dong, Tongdaemun-gu, Seoul 130-743, Korea
jaeho@uos.ac.kr

Mike Barley
Department of Computer Science
University of Auckland
Private Bag 92019, Auckland, New Zealand
barley@cs.auckland.ac.nz

Local Organization Co-chairs

Ho-Sang Ham
Internet Computing Research Department
Electronics and Telecommunications Research Institute (ETRI), Korea
hsham@etri.re.kr

Jung-Jin Yang
Department of Computer Science and Engineering
Catholic University of Korea, Korea
jungjin@catholic.ac.kr

Local Organization Committee

Soowon Lee (Soongsil University)
Suhn Beom Kwon (Dankook University)
Hyun Kim (Electronics and Telecommunications Research Institute)
Seung Ik Baek (Hanyang University)

Program Committee

Cristiano Castelfranchi (Italy)
Brahim Chaib-draa (Canada)
Joongmin Choi (Korea)
John Debenham (Australia)
Klaus Fisher (Germany)
Chun-Nan Hsu (Taiwan)
Michael Huhns (USA)
Toru Ishida (Japan)
Ilkon Kim (Korea)
In-Cheol Kim (Korea)
Minkoo Kim (Korea)
David Kinny (Australia)
Yasuhiko Kitamura (Japan)
Kazuhiro Kuwabara (Japan)
Jimmy H.M. Lee (China)
Ho-fung Leung (China)
Chao-Lin Liu (Taiwan)
Jiming Liu (China)
Jyi-shane Liu (Taiwan)
Rey-long Liu (Taiwan)
Jian Lu (China)
Michael Luck (UK)
Xudong Luo (UK)
John Jules Meyer (The Netherlands)
Luc Moreau (UK)
Joerg Mueller (Germany)
Hideyuki Nakashima (Japan)
Ei-Ichi Osawa (Japan)
Ichiro Osawa (Japan)
Sascha Ossowski (Spain)
Young-Tack Park (Korea)
Van Parunak (USA)
Ramakoti Sadananda (Thailand)
Zhongzhi Shi (China)
Wookho Son (Korea)
Liz Sonenberg (Australia)
Von-Wun Soo (Taiwan)
Toshiharu Sugawara (Japan)
Ron Sun (USA)
Qijia Tian (China)
Jung-Jin Yang (Korea)
Makoto Yokoo (Japan)
Xinghuo Yu (Australia)
Soe-Tsyr Yuan (Taiwan)
Chengqi Zhang (Australia)

Table of Contents

A Multi-agent Approach to Business Processes Management in an Electronic Market

Boon-Hua Ooi

Faculty of IT, University of Technology, Sydney
PO Box 123, NSW 2007, Australia
booi@it.uts.edu.au

Abstract. The world's industries are moving into the 'new economy' of electronic market. The success of this move depends on efficient and reliable management of the business processes that will drive the e-Market's value-chain. Efficiency can only be achieved through automation. Reliability will be achieved by reliable and dependable process management systems. These electronic trans-corporate processes cover the range from basic business transactions, workflows to sophisticated business processes that rely on a high level of personalisation. Management of these processes requires safety critical computer systems that can survive local catastrophes. Multi-agent systems have this level of dependability and personalisation. An analysis of the full range of processes is followed by an implementation framework for their management using a powerful multiagent architecture.

1 Introduction

The new economy of electronic market is creating demands for a new generation of Internet applications that can dramatically automate *trans-corporate industry processes* only if the business systems and data that drive these processes are integrated across the component organisations [1]. Here *trans-corporate industry processes* include both business transactions, and business processes and workflows. These Internet applications can only automate these processes if there is a method to manage collaborative processes across organisations and to provide data interoperability. Improvements in process management can only be achieved through automation. Automation of processes leads to faster cycle times, reduced overhead and more competitive offerings. *Industry process re-engineering* is the re-engineering of trans-corporate processes as electronically managed processes. Companies that have implemented this e-business vision are saving tens of millions of dollars per year [2]. Industry process re-engineering must address the four issues of complexity, interoperability, communication and management:

- *complexity* of trans-corporate processes refers to their nature which includes *all* trans-corporate processes from routine workflows to high-level emergent processes [3];
- *interoperability* is an issue due to the heterogeneity of the diverse distributed systems across a trading community. These systems vary in the applications that manage them and in the data formats that they employ;

J. Lee and M. Barley (Eds.): PRIMA 2003, LNAI 2891, pp. 1–12, 2003.

- the *communication* and messaging infrastructure chosen will operate in a mission-critical distributed environment, and
- process *management* which is responsible for tracking the automated trans-corporate processes that may include processes unique to individual trading partners and will probably involve a wide range of process steps that must "make sense" to all involved.

That is, industry process re-engineering must deliver a secure, scalable and reliable solution for running a company's most critical core business processes. The complex nature of these processes is considered here.

Trans-corporate processes range in type from production workflows to high-level emergent processes. Processes all across this range are to be managed in the distributed and diverse e-business environment. High-level emergent processes are business processes that are not predefined and are ad hoc. These processes typically take place at the higher levels of organisations [1], and are distinct from production workflows [2]. Emergent processes are opportunistic in nature whereas production workflows are routines. How an emergent process will terminate may not be known until the process is well advanced. Further, the tasks involved in an emergent process are typically not predefined and *emerge* as the process develops. Those tasks may be carried out by collaborative groups as well as by individuals [4]. For example, in a manufacturing organisation an emergent process could be triggered by "lets consider introducing a new product line for the US market".

From a process management perspective, emergent processes contain "knowledge-driven" sub-processes and conventional "goal-driven" sub-processes [5]. The management of a *knowledge-driven process* is guided by its 'process knowledge' and 'performance knowledge' and *not* by its goal which may not be fixed and may mutate. On the other hand, the management of a *goal-driven process* is guided by its goal which is fixed, although the individual corporations involved in a trans-corporate process may not achieve such a fixed goal in the same way.

Multiagent technology is an attractive basis for industry process re-engineering [6]. A multiagent system consists of autonomous components that interact with messages. The scalability issue is "solved" – in theory – by establishing a common understanding for inter-agent communication and interaction. Specifying an inter-agent communication protocol may be tedious but is not technically complex. Standard XML-based ontologies will enable data to be communicated freely [1] but much work has yet to be done on standards for communicating expertise. Specifying the agent interaction protocol is a more complex as it in effect specifies the common understanding on the basis of which the whole system will operate. A multiagent system to manage "goal-driven" processes is described in [7]. In that system each human user is assisted by an agent which is based on a generic three-layer, BDI hybrid agent architecture. The term *individual* refers to a user/agent pair. That system has been extended to support knowledge-driven processes and so to support emergent process management and the full range of trans-corporate processes. The general business of managing knowledge-driven sub-processes is illustrated in Fig. 1. Any process management system should address the "process knowledge" and the "performance knowledge. *Process knowledge* is the wisdom that has been accumulated, particularly that which is relevant to the process instance at hand. *Performance knowledge* is knowledge of how effective people, methods and plans are at achieving various things. Sec. 3 describes the agents' reasoning mechanisms. Sec. 4

discusses the management of the process knowledge. Sec. 5 describes the implementation of process management agents using JACK, its distributed architecture and interaction mechanism offer through JACK's *dci* network.

2 Trans-corporate Processes

Following [8] a *business process* is "a set of one or more linked procedures or activities which collectively realise a business objective or policy goal, normally within the context of an organisational structure defining functional roles and relationships". Implicit in this definition is the idea that a process may be repeatedly decomposed into linked sub-processes until those sub-processes are "activities" which are atomic pieces of work. [viz (op.cit) "An *activity* is a description of a piece of work that forms one logical step within a process."]. A particular process is called a (process) *instance*. An instance may require that certain things should be done; such things are called *tasks*. A *trigger* is an event that leads to the creation of an instance. The *goal* of an instance is a state that the instance is trying to achieve. The *termination condition* of an instance is a condition which if satisfied during the life of an instance causes that instance to be destroyed whether its goal has been achieved or not. The *patron* of an instance is the individual who is responsible for managing the life of that instance [5]. At any time in a process instance's life, the *history* of that instance is the sequence of prior sub-goals and the prior sequence of knowledge inputs to the instance. The history is "knowledge of all that has happened already".

From a process management viewpoint, trans-corporate processes can be seen as consisting of sub-processes that are of one of the three following types:

- A *task-driven process* has a unique decomposition into a – possibly conditional – sequence of activities. Each of these activities has a goal and is associated with a task that "always" achieves this goal. Production workflows are typically task-driven processes.
- A *goal-driven process* has a process goal, and achievement of that goal is the termination condition for the process. The process goal may have various decompositions into sequences of sub-goals where these sub-goals are associated with (atomic) activities and so with tasks. Some of these sequences of tasks may work better than others, and there may be no way of knowing which is which [3]. A task for an activity may fail outright, or may be otherwise ineffective at achieving its goal. In other words, process failure is a feature of goal-driven processes. If a task fails then another way to achieve the process goal may be sought.
- A *knowledge-driven process* has a process goal, but the goal may be vague and may mutate [9]. The process patron, often in the light of knowledge generated during the process, determines the mutations. After performing a task in a knowledge-driven process, the "next goal" is chosen by the process patron. This choice is made using general knowledge concerning the process – called the *process knowledge*. The process patron then chooses the tasks to achieve that next goal. This choice may be made using general knowledge about the effectiveness of tasks – called the *performance knowledge*. So in so far as the process goal gives

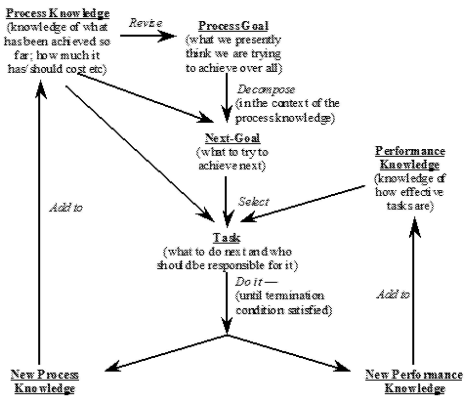


Fig. 1. Knowledge-driven process management

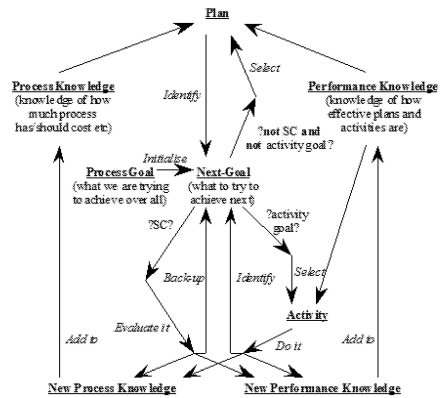


Fig. 2. Goal-driven process management

direction to goal-driven – and task-driven – processes, the growing body of process knowledge gives direction to knowledge-driven processes. The management of knowledge-driven processes is considerably more complex than the other two classes of process, see Fig. 1. But, knowledge-driven processes are "not all bad" – they typically have goal-driven sub-processes.

In contrast to Fig. 1, Fig. 2 shows a simplified view of the management of goal-driven process. The primitives shown in Fig. 2 are goals and plans. As the successful execution of a plan for a goal-driven process is not necessarily related to the achievement of the process goal each plan ends with a "success condition". A plan's *success condition* (SC) is a procedure that determines whether the plan's goal has been achieved. Some goals are associated with executable procedures. If a goal is not associated with an executable procedure then it should be the subject of at least one plan. The ideas shown in Fig. 2 are discussed in greater detail below. Figure 2 presents a simplified view because a sub-goal of a goal-driven goal will not necessarily be goal-driven, and because plans as shown there are assumed not to be aborted.

Task-driven processes may be managed by a simple reactive agent architecture based on event-condition-action rules [5]. Goal-driven processes may be modelled as state and activity charts [10] and managed by plans that can accommodate failure [11]. Such a planning system may provide the deliberative reasoning mechanism in a BDI agent architecture [11] and is used in a goal-driven process management system [7] where tasks are represented as plans for goal-driven processes. But the success of execution of a plan for a goal-driven process is not necessarily related to the achievement of its goal. One reason for this is that an instance may make progress outside the process management system – two players could go for lunch for example. So each plan for a goal-driven process should terminate with a check of whether its goal has been achieved.

3 Reasoning

The conceptual architecture of the process agents described here is a 3-layer, BDI hybrid architecture. As a hybrid architecture the process agent architecture exhibits both deliberative and reactive reasoning [12]. Deliberative reasoning is managed within a belief-goal-plan-intention framework [11]. Reactive reasoning is effected with triggers. Theses two forms of reasoning are balanced by giving reactive reasoning precedence over deliberative reasoning. That is, an attempt is made to fire all reactive triggers before each deliberative cycle commences.

3.1 Deliberative Reasoning

The process agent employs a form of "plan" that is rather more elaborate than many types of agent plan [11]. For goal-driven process management plans are prone to local failure. A form of plan that can deal naturally with failure is described in [op. cit.]. Plans are built there from single-entry, triple-exit blocks; where the three exits represent success, failure and abort. Powerful though that approach is, it is insufficient for goal-driven processes because whether a plan has executed successfully is not necessarily related to whether that plan's goal has been achieved.

A plan cannot necessarily be relied upon to achieve its goal even if all of the sub-goals on a chosen path through the plan have been achieved. On the other hand, if a plan has failed to execute then it is possible that the plan's goal may still have been achieved. So, in business process applications a necessary sub-goal in every plan body is a sub-goal called the "success condition". The *success condition* (SC) is a procedure whose goal is to determine whether the plan's goal has been achieved. The success condition is the final sub-goal on *every* path through a plan. The success condition is a procedure; the execution of that procedure may succeed (✓), fail (✗) or abort (**A**). If the execution of the success condition fails then the overall success of the plan is unknown (**?**). So the four possible plan exits resulting from a plan are success (✓), fail (✗), abort (**A**) and unknown (**?**).

Goal-directed business processes may be modelled as state and activity charts [10]. The primitives of that model are activities and states. Emergent sub-processes are seldom modelled. For goal-driven sub-processes the *states* correspond to the goals associated with that sub-process. An *activity chart* specifies the data flow between activities. An activity chart is a directed graph in which the arcs are annotated with data items. A *state chart* is a representation of a finite state machine in which the transitions annotated with Event-Condition-Action rules. If the transition from state X to state Y is annotated with E[C]/A then this transition is followed if event E occurs and condition C holds in which case action A is performed. If business processes are one-of-a-kind then they are not modelled in advance. A *plan library* supports one-of-a-kind sub-processes, see Fig. 1. To support emergent sub-processes an *activity library* is embedded in the performance knowledge base. These two libraries provide a repertoire of building blocks that may be assembled to support a strategic business process.

The deliberative frequency is the frequency at which the deliberative process is activated. This process determines current options, selects current goals, and so on as described in the agent control architecture above. The deliberative frequency should

be short enough to keep the system moving but not so fast that an individual's "In Tray" is seen to be constantly changing. For general business process management a deliberative frequency in the region of 15–60 minutes seems appropriate.

3.2 Reactive Reasoning

Reactive reasoning play three roles: first, a plan is aborted if its specified abort condition is satisfied, second, data is passed to partly executed plans for goals an agent is committed to achieve, and third, urgent messages are dealt with. Of these three roles the third takes precedence over the second. For example, the third role for reactive triggers handles urgent messages of the form "stop what you are doing and do this"; this third role has yet to be implemented.

Each plan contains an optional abort condition [ab] as described above. These abort conditions are realised as *abort triggers*. These triggers scan the agent's beliefs for the presence or absence of specific conditions. For example, "*if* I do not believe that X wants Y *then abort* the plan whose goal is to deliver Y to X" is an example of an abort trigger. Abort triggers are only active if the goal of the plan to which they are attached is a goal that the agent is presently committed to achieving. If a plan is aborted then any active sub-goals of that plan are also aborted.

Data is passed to partly executed plans using reactive triggers. For example, the goal of the plan illustrated in Fig. 3 is "X's opinion on Y has been obtained". The plan for that goal has one sub-goal ["X's opinion on Y" requested] and another sub-goal ["X's opinion on Y is Z" asserted]. This second sub-goal may be achieved if "X's opinion on Y is Z" is present in the agent's world beliefs. So until such a belief materialises an attempt to achieve this second sub-goal may "hang". This situation is managed by linking the second sub-goal to a reactive trigger "I believe that: X's opinion on Y is...". This reactive trigger "watches" the agent's world beliefs. If and when this reactive trigger fires the second sub-goal is instantiated and is achieved. Reactive triggers of this form are associated with sub-goals. These triggers are activated when their associated sub-goal is committed to but has not been achieved. Triggers of this form provide a mechanism for passing data derived from inter-agent communication to such sub-goals.

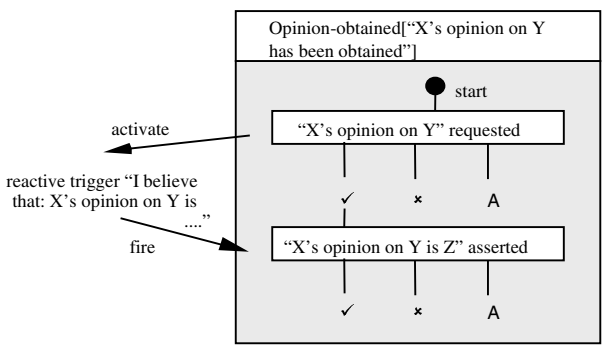


Fig. 3. Segment of a plan and a reactive trigger

The reactive frequency is the frequency at which an attempt is made to fire all active abort triggers and reactive triggers. The reactive frequency should be fast enough to avoid delays but not so fast that the agent is repeatedly scanning a set of seldom changing beliefs. The reactive frequency is set at the order of one minute. So the reactive frequency is shorter than the deliberative frequency. The abort triggers have a higher priority than reactive triggers. So if a plan's abort trigger fires and if an active sub-goal in that plan is the subject of a reactive trigger then that sub-goal will be deactivated so preventing that reactive trigger from firing even if the required belief is in the world beliefs.

4 Process Knowledge and Goal

Process knowledge is the wisdom that has been accumulated, particularly that which is relevant to the process instance at hand. For knowledge-driven processes the management of the process knowledge is shown on the left-hand side of Fig. 1. For knowledge-driven processes, management of the process knowledge is impractical.

The process knowledge in any real application includes an enormous amount of general and common sense knowledge. For example, the process trigger "the time is right to look at the US market" may be based on a large quantity of empirical knowledge and a fund of experiential knowledge. So the system does not attempt to represent the process knowledge in any way; it is seen to be largely in the heads of the users. The system does assist in the maintenance of the process knowledge by ensuring that any virtual documents generated during an activity in a knowledge-driven sub-process are passed to the process patron when the activity is complete. Virtual documents are either interactive web documents or workspaces in the LiveNet workspace system that is used to handle virtual meetings and discussions.

The system records, but does not attempt to understand the process goal. The patron carries out any possible revisions of the process goal without assistance from the system. Likewise, the decomposition of the process goal to decide "what to do next" – the next-goal. It may appear that the system does not do very much at all! If the next-goal is the goal of a goal-driven process – which it may well be – then the system may be left to manage it as long as it has plans in its plan library to achieve that next-goal. If the system does not have plans to achieve such a goal then the user may be able to quickly assemble such a plan from existing components in the plan library. The organisation of the plan library is a free form, hierarchic filing system designed completely by each user. Such a plan only specifies what has to be done at the host agent. If a plan sends something to another agent with a sub-goal attached it is up to that other agent to design a plan to deal with that sub-goal. If the next-goal is the goal of a knowledge-driven process then the procedure illustrated in Fig. 1 commences at the level of that goal.

So for this part of the procedure, the agent provides assistance with updating the process knowledge, and if a next-goal is the goal of a goal-driven sub-process then the system will manage that sub-process, perhaps after being given a plan to do so.

5 Implementation

The transformation of the conceptual model of process agents into deployable architecture requires the following:

- Availability of "foundation" framework that provides and supports basic (core) agent-oriented behaviours.
- Facility that allows high-level declaration of specific *distributed application architecture* for the process agents and set-up that *enables remote communication* between *distributed* process agents.
- A tool that enables creation of *agent constructs* and component process agents.

The essential programming constructs in the process agent architecture are: agent's *goal*, *event* (trigger), *plan* (task) and *belief*. To be a genuine agent architecture, the implementation framework must allow these constructs to be programmed declaratively.

5.1 The Implementation Technology

A plausible implementation strategy would be employing an agent development tool that can formulate an "interaction map" depicting workflows of the process agents. An approach to construct such a practicable process management computing framework is to leverage on the agent-oriented support offer by JACK Intelligent Agents™. JACK is an *Agent Oriented* development environment built on top of Java programming language. It offers specific agent-oriented API extensions to implement agent behaviours. JACK models the reasoning behaviour according to the **B**elief **D**esire **I**ntention (**BDI**) model of artificial intelligence.

Conforming to the BDI agent model, JACK agents are autonomous software components that have explicit goals to achieve (desires) or events to handle (triggers). To describe how they should go about achieving these desires, these agents are programmed with a set of plans ("tasks"). Each plan describes how to achieve a goal or respond to a trigger (internal or external) under varying circumstances dictated predominantly by the current 'mental state' of an agent executing the relevant plan. A plan is made up of a sequence of *atomic actions* that collectively represent the "task" needed to be performed. Set to work, the agent reacts or pursues its given goals (desires), adopting the appropriate plans (intentions) according to its current set of data (beliefs) about the state of the world. This combination of desires and beliefs initiating context-sensitive intended behaviour is part of what characterises a BDI agent.

A very significant feature about thread handling in JACK is to draw on the concept of *Finite State Machine* (FSM). As a multi-agent computing framework (which is a JAVA based framework) always involves in continuously spawning a large number of processing threads, and most of the threads will have the need to access common objects. Thus switching between thread executions, which directly affecting object locking and concurrency control have became issues that demand meticulous attention. In JACK, such multi-threading issues are taken care of and made simpler through the use of FSM statements. FSM statements enable a task execution to be segregated into a number of indivisible atomic steps (FSM steps). The JACK kernel

insures that each FSM step is *atomic* and thus ensuring "*safe switching*" between multiple threads. The multi-agent framework for a heavy-duty process management engine could easily generate hundreds of execution threads during runtime. Such complex thread-intensive execution environment will be prevailed with multiple process threads that need to access common beliefs of an agent in order to make "informed" decision and subsequently achieve any goal adopted. Such framework benefits tremendously from the guaranteed safe points at which execution thread can be switched (because reasoning methods and task executions in JACK is FSMs). With so many running threads exist in the memory at any point of time, it is also important that a carefully planned *exit strategy* should be coded in order to avoid any race condition that will eventually leads to unpredictable end result.

JACK provides networked runtime environments (supported by a communication layer known as *dci* network) at distributed locations (distributed computing machines) on which different agent processes are able to operate. These *distributed agents* address communication messages (MessageEvent and BDIMessageEvent) to one another by specifying the name of the destination agent and its resident host, which is known as agent process *portal*. For instance, a remote agent will be addressed as *agent @ portal*. A set of portal can be connected explicitly (directly) or a name-server can be used so that portals can be located and connected on the fly as needed. By default, the transporting mechanism of JACK's *dci* network is the UDP transport protocol available on all TCP/IP networks. However UDP transport protocol is connectionless and is unreliable, thus JACK's *dci* network includes a thin layer on top of UDP which guarantees reliable peer to peer communications.

5.2 The Deployment Specifications

Constructions of process management agents (PMA) using JACK primarily amount to declaratively specify the required component agents that collectively form a trans-corporate process management framework. With respect to this technique, the reasoning behaviours of PMA can be categorized under two broad categories: *pro-active* and *reactive*. In pro-active behaviour, the process agent is constantly being reminded of its intended objective. Under such circumstances, the 'stimulus' of a PMA appears in the form of a committed *goal* (desire) and the agent's action is driven by goal directed responses. On the other hand, reactive behaviour is spontaneous and is *event* driven.

A *basic* JACK process management agent is made up of:

- A set of beliefs about the world (its data set);
- A set of events that it will respond to;
- A set of goals that it may desire to achieve
- (either at the request of an external entity such as a user or another process agent, as a consequence of an event, or when one or more of its beliefs change); and
- A set of plans that describe how it can handle the goals or events that may arise.

Figure 4 shows a typical BDI agent-oriented application framework created using an IDE that supports declarative programming of JACK Intelligent Agents™. The four essential BDI constructs: goal, event, plan and belief of JACK can be visually

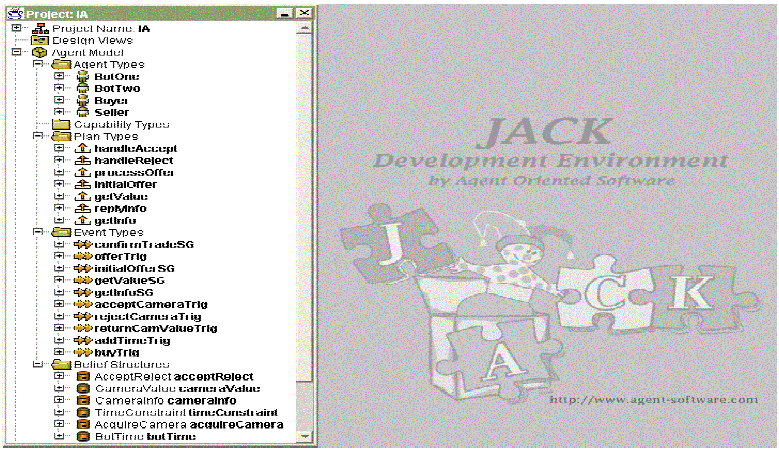


Fig. 4. Typical BDI framework of JACK agents (IDE of JACK Intelligent Agents™)

declared and defined. Throughout the process of agent construction, one of the primary tasks is to declare all the relevant events for the particular application architecture. These declared events would serve as the trigger point of all reasoning behaviours of a PMA. Different agent's event (which serves different purpose) can be created by simply select from a variety of event types or classes that are available. Discussion here will focus on the BDI events as they are the main building blocks that form the reasoning behaviours of an agent.

Deliberative reasoning: A committed goal would model the deliberation of a PMA by committing to a desired outcome. In JACK, this commitment is achieved through the use of triggering event that extends BDIGoalEvent (a Java base class for events which offers all the BDI features by default). The BDIGoalEvent represents a goal or an objective that an agent wishes to achieve and choose to undertake a task to realise the goal. The "state of mind" of a pro-active PMA after a BDIGoalEvent has been adopted would be:

- *Achieve* the adopted goal;
- *Insist* that the goal is achieved, by ensuring that the goal has been met when the task is completed;
- *Test* whether the goal can be achieved using *success condition* (SC) procedures; or
- *Determine* a situation in which the goal can be achieved.

Deliberative cycle: Within a PMA agent, this is the repeating loop in which a committed goal is reviewed and evaluated in order to determine its validity or its completion, i.e. whether the goal is still a valid one needs to be achieved or the goal has been expired. In such event, the committed goal appears to be a slightly "longer term" interest. The use of BDIGoalEvent approach is improvised with an additional belief goal, which in this case is a *belief set* that implements an interface – GoalItf (a simple Java markup interface). The goal "reviewing" process or the deliberative cycle is implemented through the use of *change cursor* that monitors changes occur in belief goal. The entire deliberative cycle is made complete by having the *change cursor* work together with a @wait_for() reasoning statement provided in JACK. A PMA

agent with capability of performing deliberative cycle will be attributed with one or more belief goal, it is then equipped with plan that activates an execution loop that is incorporated with a @wait_for() statement that move the plan into the appropriate action as soon as the *change cursor* detects changes has taken place in the belief goal.

Reactive reasoning: The reactive capabilities of PMA is implemented by event extending any of the 2 Java base classes: BDIFactEvent and BDIMessageEvent. The 2 base classes enable *meta-level reasoning* to be performed to select the most appropriate applicable plan instance for execution in respond to a reactive trigger. Once an intended means has been chosen, it will be committed. If the intended means succeeds, the BDIFactEvent or the BDIMessageEvent succeeds and execution returns to the caller that initiates the trigger. If the intended means fails, the BDIFactEvent or the BDIMessageEvent fails and execution returns to the event initiator.

Whether a deliberative goal or a reactive goal is "triggered", the PMA's response would be deciding to undertake a task to satisfy the objective of the goal. The task will be the partially instantiated plan (intended means) generated by running a *logical query* on the belief set of the agent. In the context of JACK, logical query is a method that queries the belief set and attempts to unify logical variables declared within a plan to derive at a set of applicable plan instances. Such logical query appears in the form of a logical expression in the *context()* method of a plan and will be used to evaluate an agent's beliefs of the world at a particular instant of time. A logical expression can be viewed as an expression composed of logical variables and belief set query (in JACK, belief set query is defined when an agent belief members are declared).

5.3 Agent-Oriented Development Using JACK

Developing agent-oriented software using JACK is a challenging experience and an exciting endeavour. JACK is armed with an IDE that improves coding productivity and ensures correctness in JACK's program structures. The learning curve for JACK varies according to individuals and this depends primarily on 2 bodies of foundation knowledge that an individual has acquired. Firstly, a developer with a thorough understanding of *agent architecture* based on *Belief-Desire-Intention* (BDI) framework will benefit immensely as JACK agents are assembled using the BDI constructs. The correct execution-time workflow and interaction between agents require sound knowledge on communication protocol (via internal or external message events) used in BDI framework. Secondly, on the aspect of agent codes, proficiency in JAVA programming language is a definite advantage as the final program codes generated by JACK's IDE are JAVA programs whereby all the basic agent constructs appear in the form of JAVA classes. A workable JACK multi-agent software system always requires additional program modules or methods that provide the intricate utilities and functional tasks. Such programs are incorporated into JACK agents via JAVA methods or implemented JAVA interfaces. Issues pertaining to timing of agent instantiation and invocation of critical methods based on conditional checking (in order to produce the desire agent behaviours) need an in-depth understanding of agent workflow in a multi-thread environment. A good working knowledge of JAVA will come in handy to produce effective "seamless" codes that support efficient execution performance.

6 Conclusion

Managing trans-corporate industry processes involves managing processes across the full process spectrum in the distributed e-business environment. This spectrum is analysed as processes of three distinct types [5]. The management of knowledge-driven processes is not widely understood and has been described here. A multi-agent system manages goal-driven processes and supports the management of knowledge-driven processes [7]. The conceptual agent architecture is a three-layer BDI, hybrid architecture. An implementation of such architecture can be 'realised' through JACK Intelligent Agents™. During a process instance the responsibility for sub-processes may be delegated, and possibly out-sourced in the distributed, e-business environment. To achieve this, the system forms a view on who should be asked to do what at each step in a process, and tracks the resulting delegations of process responsibility. This includes tracking delegations across component organisations.

References

[1] Skinstad, R., "Business process integration through XML". In proceedings XML Europe 2000, Paris, 12–16 June 2000.
[2] Feldman, S., "Technology Trends and Drivers and a Vision of the Future of e-business." In proceedings 4th International Enterprise Distributed Object Computing Conference, September 25–28, 2000, Makuhari, Japan.
[3] Dourish, P., "Using Metalevel Techniques in a Flexible Toolkit for CSCW Applications." ACM Transactions on Computer-Human Interaction,Vol. 5, No. 2, June, 1998, pp. 109–155.
[4] A.P. Sheth, D. Georgakopoulos, S. Joosten, M. Rusinkiewicz, W. Scacchi, J.C. Wileden, and A.L. Wolf, "Report from the NSF workshop on workflow and process automation in information systems." SIGMOD Record, 25(4):55–67, December 1996.
[5] Debenham, J.K. "Supporting Strategic Process", in proceedings Fifth International Conference on The Practical Application of Intelligent Agents and Multi-Agents PAAM2000, Manchester UK, April 2000.
[6] Jain, A.K., Aparicio, M. and Singh, M.P. "Agents for Process Coherence in Virtual Enterprises" in Communications of the ACM, Volume 42, No 3, March 1999, pp. 62–69.
[7] J.K. Debenham, "Who does what in a multiagent system for emergent process management", in proceedings 9th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems (ECBS), April 8–11 2002, Lund, Sweden, pp. 35–40.
[8] Fischer, L. (Ed.), "Workflow Handbook 2001." Future Strategies, 2000.
[9] Debenham, J.K., "Knowledge Engineering: Unifying Knowledge Base and Database Design", Springer-Verlag, 1998.
[10] Muth, P., Wodtke, D., Weissenfels, J., Kotz D.A. and Weikum, G., "From Centralized Workflow Specification to Distributed Workflow Execution." In Journal of Intelligent Information Systems (JIIS), Kluwer Academic Publishers, Vol. 10, No. 2, 1998.
[11] Rao, A.S. and Georgeff, M.P., "BDI Agents: From Theory to Practice", in proceedings First International Conference on Multi-Agent Systems (ICMAS–95), San Francisco, USA, pp. 312–319.
[12] Weiss, G. (ed)., Multi-Agent Systems. The MIT Press: Cambridge, MA (1999).

Modeling of a Multi-agent System for Coordination of Supply Chains with Complexity and Uncertainty

Hyung Jun Ahn and Sung Joo Park

Graduate School Of Management, KAIST, 207-43, Cheongryang-ni,
Dongdaemoon-gu, Seoul, Korea
{s_hjahn,sjpark}@kgsm.kaist.ac.kr

Abstract. In supply chain management, improving the efficiency of the overall supply chain is important. It has been found that sharing information between supply chain members can increase the efficiency of the supply chain by varying degrees in different situations. There are, however, many practical difficulties in sharing information with all the companies in a complex supply chain. This paper presents a multi-agent system for supply chain coordination where agents discover the structure of supply chains and share information on inventory levels only by local collaborations. Agents use the gathered information to estimate demands propagated from multiple markets through a complex network of companies with different lead times and market shares. The performance of the suggested system is investigated with a simulation experiment and the result is compared with an alternative strategy.

1 Introduction

The importance of supply chain is increasing with market globalization and the advancement of electronic commerce [17][19]. A supply chain can be defined as a network consisting of suppliers, warehouses, manufacturers, wholesalers, and retailers through which material and products are acquired, transformed, and delivered to consumers in markets. In today's supply chains, it is a critical issue to enhance the efficiency of supply chains in the perspective of the whole chains.

One of the issues many companies and researchers are concerned is how information sharing between companies can be facilitated to improve the efficiency of supply chains. For example, there have been studies that showed sharing inventory levels of buyers helps predicting future demands more accurately, resulting in lower average inventory levels and higher service rates [1][12]. But information sharing between companies is not always possible [2][7]. Supply chains can be complex and dynamic, and many companies may not want to share their information for various reasons. Moreover, even for companies who are willing to share information, incompatibility among heterogeneous information systems can hinder the information sharing. In general, it is especially difficult for distantly located companies in supply chains to share information, compared to closely located companies.

This research, reflecting these practical barriers, suggests a multi-agent system for distributed and collaborative supply chain management. Multi-agent technology has many beneficial features for autonomous, collaborative, and intelligent systems in

J. Lee and M. Barley (Eds.): PRIMA 2003, LNAI 2891, pp. 13–24, 2003.

distributed environments, which makes it one of the best candidates for complex supply chain management [20][22][25]. The suggested system of this paper was designed to improve the efficiency of supply chains by utilizing only *local information sharing* between neighboring companies. With the suggested architecture, agent systems will discover the structure of supply chains by local information sharing and utilize the structure to accurately predict future demands that will come to their companies in a specific point of time. To evaluate the performance of the suggested architecture, a simulation experiment was performed.

The remainder of this paper is organized as follows: The second section reviews related research. The third section presents the architecture of the suggested system and analyzes the performance of the system with simulation. The fourth session presents a discussion and conclusion.

2 Review of Related Research

2.1 Information Sharing in Supply Chains

There have been many efforts at showing how information sharing in supply chains can increase efficiency and reduce costs. They deal with information sharing strategies mainly involving inventory levels, buyers' strategy, or market demands; they show how the strategies affect supply chain performance in different situations using analytic methods or simulation experiments [1][3][12][9].

In contrast, there are some studies that showed sharing information can provide little benefit. The studies of [10][11] claim that investing in physical flows and utilizing history data intelligently can be enough. These contradicting results imply that the benefit of sharing information can vary with circumstances and that it is important to utilize given information carefully and effectively. For example, [3] showed that the benefit of sharing inventory information is different under diverse demand patterns and companies' processing capacities, and [5] showed that the role of the participants in a supply chain portal significantly affects the benefits of information sharing.

We also have to consider some practical barriers to information sharing. It is difficult to share information between companies using different information systems; many companies are not willing to disclose their information, worried with possible strategic losses [7]. In general, information sharing is more difficult between distantly located companies in supply chain topology than between neighboring companies. It has been actually found that some companies can incur losses owing to information sharing under certain conditions [2]. From these results, it becomes obvious that it is impractical to assume global information sharing between companies in supply chains, and also that it is difficult to effectively utilize shared information.

Another limitation in the existing researches on information sharing is that most of them used simple supply chain models, consisting of only two or three serial layers. The simplicity in the models makes it difficult to generalize their implications to practical and complex supply chain environments. This paper will try to overcome this limitation, while considering the practical difficulties of information sharing.

2.2 Supply Chain Management and Multi-agent Technology

There are unique characteristics required for information systems that support supply chain management. First, they should be able to support distributed collaboration among companies. Second, collaborations in supply chains cannot be governed by a single company in a one-directional way, but needs to be coordinated by autonomous participation of companies. Third, they need a high level of intelligence for planning, scheduling, and change adaptation. For these reasons, agent technology is regarded as one of the best candidates for supply chain management [20][22][14][25].

Studies on agent-based supply chain management can be classified into three categories. The first type of research is concerned with the coordination aspect. In this type of research, various types of companies and their capabilities are modeled into individual agents and their interactions are designed for efficient collaboration [21][16][23]. The second type of research focuses on simulation of supply chains using agent-based models. This type of research tries to discover the performance of agent-based supply chain architectures under various strategies and constraints [16][18][22]. The third type of research studies how virtual supply chains can be organized flexibly by multi-agent systems [15][24]. For example, [15] showed how virtual supply chains can be formed by solving distributed constraint satisfaction problems by agents.

The studies listed above provide good basis for modeling various aspects of supply chains. They are, however, deficient in showing how complex supply chains can be coordinated under the practical difficulties of information sharing discussed earlier. Many of them assume that global information sharing is possible and/or use simple supply chain models. In order to derive more practical implications for complex supply chains, this paper presents a multi-agent system architecture incorporating more realistic assumptions regarding the practical difficulties.

3 MADC: A Multi-agent System for Distributed Coordination of Supply Chains

3.1 Summary of the Problems

Figure 1 illustrates the problems that this paper focuses on. A supply chain can produce products for multiple markets. Also, an individual company is likely to have only limited visibility of the supply chain structure, which makes it difficult to make future demands estimations, because the pattern of demand propagation through the supply chain depends on the capabilities and strategies of companies along the path from the markets to the company. These problems are further amplified if the supply chain can change over time dynamically. As a result of these problems, individual companies are likely to make inaccurate demand estimations and the supply chain can suffer from the well-known Bullwhip effect [6][13]. The Bullwhip effect refers to the problem where the fluctuations of productions and inventory levels are amplified in the upstream parts of supply chains than in the downstream parts. The Bullwhip effect significantly increases operation and production costs and drops service levels.

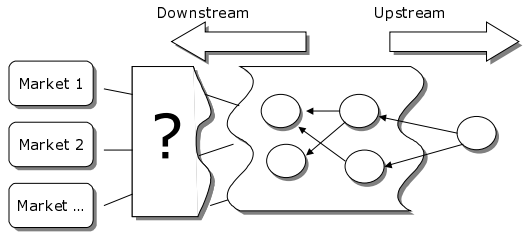


Fig. 1. The problems of complexity, information sharing, and uncertainty

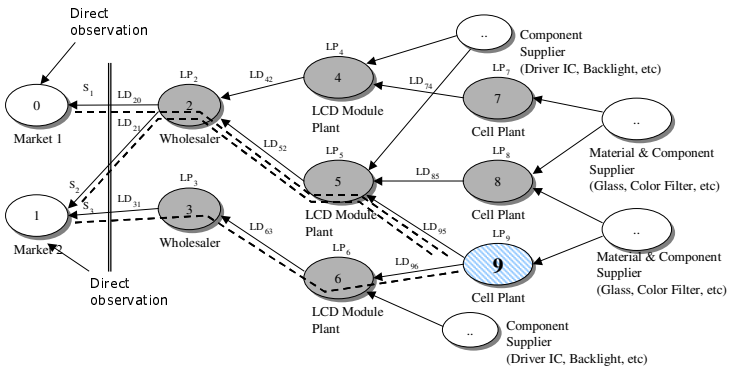


Fig. 2. An example supply chain model: a TFT-LCD supply chain of a Korean company

Table 1. Notations in the example supply chain

Notation	Description
LP_i	Lead time for production or processing at node i
LD_{ij}	Lead time for delivery from node i to node j
S_p	Market share of path p

3.2 Overview of MADC Architecture

Figure 2 shows an example supply chain model of TFT-LCD products of a Korean company. In this supply chain, there exist two distinct markets and many types of companies: suppliers of material and components, LCD cell manufacturing plants, LCD module manufacturing plants, and wholesalers. This supply chain model will be used for explaining and evaluating MADC.

Figure 2 also depicts the key approaches of MADC to solve the problems introduced in 3.1. Followings are the three key ideas of MADC:

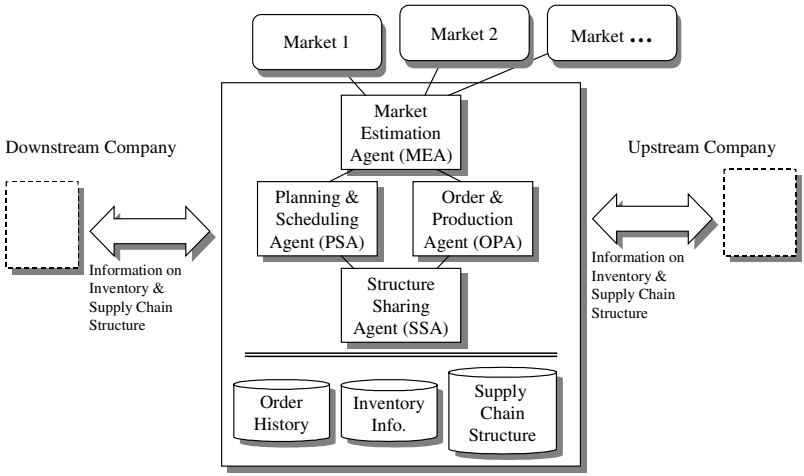


Fig. 3. The architecture of MADC

A. Sharing 'paths to markets' by local information sharing: In a complex supply chain, there are usually multiple paths between a company and consumer markets (for example, see node 9 which has three paths to markets) Considering the practical difficulties of global information sharing, MADC will share paths to markets by only local information sharing, that is, information sharing by only agents of neighboring companies. Each company will use the paths to estimate the propagation of market demands.
B. Direct observation of markets: Market demands are likely to be distorted as they are propagated to upstream parts of a supply chain. Thus, it often results in erroneous estimation to observe the demands of companies in the middle of the paths to markets. Also, there are many practical barriers to full information sharing with the companies. Thus, it can be often economic and beneficial to observe market demands directly. The observation is used together with the paths for accurate estimation.
C. Sharing inventory information of neighboring buyers: As shown in the literature review, sharing inventory information can be beneficial. In MADC, companies will have visibility of neighboring buyers' inventory levels, again with local information sharing. When inventory levels of buyers are high, suppliers can reduce production, and when the levels are low, they can increase production.

Figure 3 shows the architecture of MADC. There are four different agents in each company of a supply chain. The agents will collaborate with the agents of neighboring companies in upstream and downstream sides, while observing market demands directly.

3.3 Coordination of Agents in MADC

In MADC, agents collaborate with both internal and external agents. Thus, we need to clearly specify how the activities of the four types of agent should be coordinated in

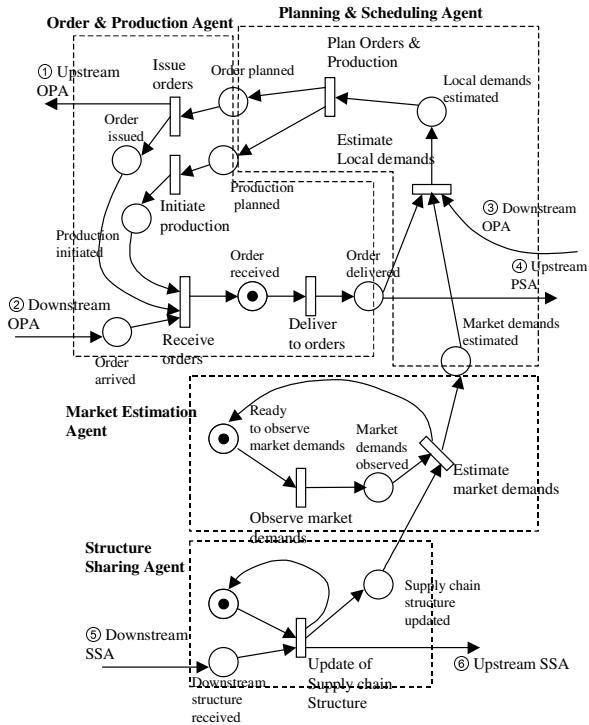


Fig. 4. Coordination of the four agents in MADC

Table 2. The conversations among agents in MADC

Name of Conversation	Protocol	Participants (Initiator → Counterpart)	Exchanged information
Query Order Plan	Query	OPA → PSA	orderPlan
Query Production Plan	Query	OPA → PSA	productionPlan
Share Inventory Level	Query	[U] PSA → [D] OPA	inventoryLevel
Issue Order Request	Request	[D] OPA → [U] OPA	order
Query Market Demands	Query	PSA → MEA	marketDemands
Query Paths to Markets	Query	MEA→ SSA	marketPath
Share Supply Chain Structure	Query	[U] SSA → [D] SSA	marketPath

harmony. This includes the identification of points where agents should share information and synchronize their executions. Figure 4 shows the coordination model of MADC with a Petri-Net that is widely used for specifying coordination of multiple independent systems when they are concurrently executed and synchronization should be considered. The numbered arrows (① ~ ⑥) represent inter-company collaborations.

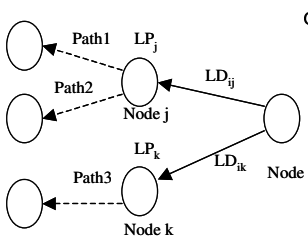
Communication among agents is performed by a set of messages that follow predefined protocols. In MADC, FIPA (The Foundation of Intelligent Physical

Agents)'s two protocols, Query and Request, were used to model the conversations among agents [26]. Table 2 shows the list of conversations in MADC, protocol types of each conversation, participants of conversations, and the type of information exchanged by each conversation.

3.4 Sharing Paths to Markets

Paths to markets are determined by the structure of supply chains. MEA uses the paths to estimate future demands incoming to its company. Demand in a market associated with a path is realized in each company after the total lead time of the path. Thus, it is sufficient to know paths to markets to estimate future demands if we assume that market demands and market share of each path can be observed directly. Moreover, paths to markets are aggregated information that can be propagated along the paths without the risk of disclosing sensitive information and the burden of communication between distantly located companies.

Figure 5 illustrates the procedure of propagating paths to markets by the collaboration of agents in neighboring companies. In node j, there are two paths to markets; in node k, there is one path. As a consequence, node i will have three paths to markets. The paths from node j will be updated by adding the production (or processing) lead time of node j and the delivery lead time between i and j. The same procedure will be applied to the path going through node k. This procedure is presented with a pseudo-code in the right side of Fig. 5. Finally, node i will use the three paths and the observed market demands to estimate future demands. Figure 6 shows an example message that is exchanged by SSAs of neighboring companies for propagating paths. In the message, two paths of node 5 in the LCD supply chain are propagated to node 8, a cell plant.




```
OBJECT SSA
 WHILE (true)
  FOR EACH b IN in the set of direct buyers
    buyerMarketPath = performConversation (b.SSA, Query, "marketPath");
    updateMarketPath (buyerMarketPath, b.leadtimeProductionProcessing,
      b.leadtimeDelivery);
  END FOR
  FOR EACH m IN the set of consumer markets
    performConversation (MEA, Query, "marketPath", m);
  END FOR
 END WHILE
END OBJECT
```

Fig. 5. Sharing paths to markets

```
(inform
  :sender LCD_module_plant_05
  :receiver cell_plant_08
  :content ( (= (all ?x (pathToMarket ?x) )
    (set (pathToMarket :totalLeadTime 8 :marketShare: 40 :endMarket market00)
         (pathToMarket :totalLeadTime 10 :marketShare: 80 :endMarket market01 ) ) ) )
  :ontology MADC_SCM
  :protocol FIPA_REQUEST
  :language FIPA_SEMANTIC_LANGUAGE
)
```

Fig. 6. An example message used for sharing paths to markets

Table 3. Notations for demand estimation

Notation	Description
S_{pi}	Market share of the retailer (final supplier) in path p of node i
DE_{ij}^{t}	Estimate of demands coming from node j to node i at time t
D'_{p}^{t}	Estimated demand at time t in the consumer market of path p based on direct observation
PM_{ij}	Set of paths to markets at node i that includes a direct buyer j

Equation (1) is used to estimate demands coming from node *j* node to *i* at time *t*, utilizing the paths to markets and direct observation. The estimation is the sum of the product of market share associated with path *p* and the market demand directly observed by MEAs after the total lead time of each path.

$$DE_{ij}^{t} = \sum_{p \in PM_{ij}} S_{pi} D'_{p}^{LM_{pi}} \tag{1}$$

3.5 Evaluation of MADC by Simulation

The performance of MADC was analyzed by a simulation experiment by comparing it with an alternative strategy using the example supply chain. In the alternative strategy (NC: No Coordination), each company estimates demands independently using past order history from buyers. For the demand pattern in the markets, we used a monotonically increasing pattern with fixed variance. Accordingly, for the alternative strategy, the double exponential smoothing (DES) technique was used for independent estimation of demands [27]. The parameters of DES were chosen from values of *.05n* (n = 0, 1, 2, … 20) that showed the best result. Followings are the assumptions for the simulation experiment:

- The cost of ordering and delivery are not considered.
- The performance of the supply chain is analyzed by two measures: inventory levels and *service rate*. Service rate is defined as *the proportion of buyers' orders that are fulfilled by suppliers*. For example, if 100 products are ordered and 90 products are actually delivered, service rate is 0.9.
- The production or processing capacity of each node is infinite.
- If shortage occurs at a supplier, it is not handled as back orders, but it is reflected in service rates.

Performance of MADC. Figure 7 shows a graph that compares the service rate of MADC with that of NC at different levels of estimation errors for market demands: 10%, 20%, and 30%. The horizontal axis represents each node of the LCD supply chain. The graph suggests that the two approaches show approximately equivalent service rates when the estimation error is around 30%. This indicates that if the estimation error is larger than 30%, using the approach of MADC can be irrational.

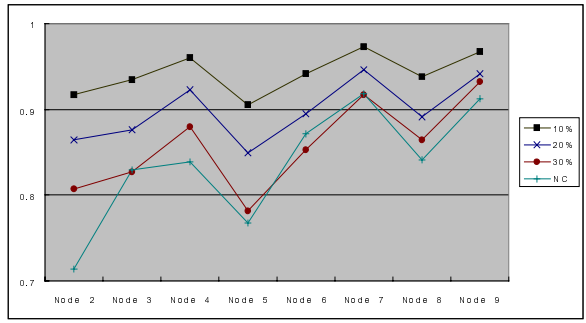


Fig. 7. Comparison of service rates between MADC and NCF

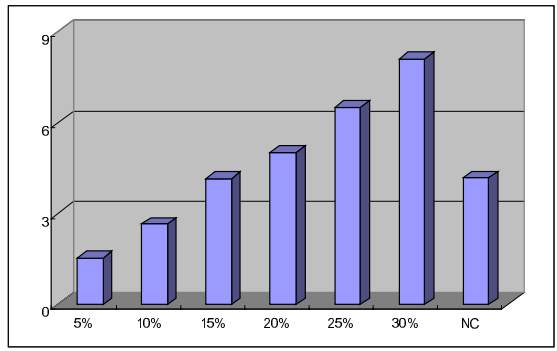


Fig. 8. Comparison of normalized inventory levels between MADC and NC

Figure 8 shows the comparison of normalized inventory levels between MADC and NC. This graph also shows inventory levels of MADC at different levels of estimation errors for market demands. The graph implies that the average inventory levels of MADC approximates that of NC when the estimation error is 15%; at larger levels of estimation errors, MADC shows higher inventory levels.

From these results, it can be concluded that the approach of MADC outperforms the independent estimation strategy only when the estimation of market demands is better than a certain point of accuracy. We can conjecture that this result comes from the strong dependency of MADC on the estimation of market demands by MEAs.

Bullwhip effects in MADC. Figure 9 shows that the Bullwhip effect is decreased as the estimation of market demands gets more accurate in MADC. For example, when the estimation error is about 30 %, there exist big difference between the inventory levels of downstream nodes and upstream nodes (e.g., compare node 2 with node 9). When the estimation error is very low, however, the bullwhip effect is significantly reduced. For example, when the estimation error is about 5%, we can notice little difference between the inventory levels of node 2 and node 9. Consequently, MADC reduces Bullwhip effects significantly when the estimation of market demands is more accurate.

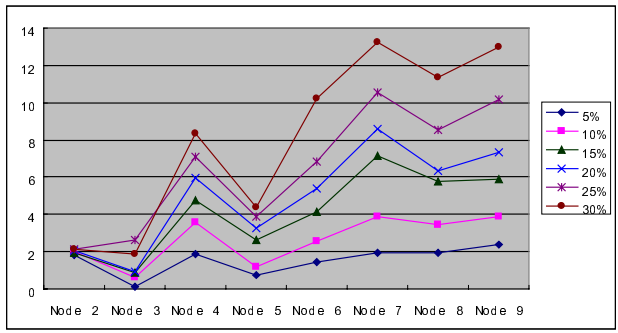


Fig. 9. Reduction of Bullwhip effects in MADC (Y axis = normalized inventory levels)

4 Discussions and Conclusion

The importance of supply chain management is increasing with globalization and the widespread adoption of electronic commerce. However, supply chains can change over time and companies in supply chains can have only limited visibility of the supply chains. This paper suggested a multi-agent system named MADC where companies can increase the efficiency of a supply chain by only local information sharing. The ideas behind the suggested system are: observing market demands directly, sharing paths to markets, and sharing inventory information among neighboring companies. The performance of the suggested system was analyzed by a simulation experiment. The result of the simulation revealed that the suggested system can show better performance when the estimation of market demands by direct observation is more accurate than a certain point. If the estimation error is large, the suggested approach can show inferior results compared to the independent estimation strategy that utilizes past order history.

There are some limitations in this research. First, if we cannot observe market demands directly, the approach cannot be used. In practice, however, the demands of consumer markets are often available, while the information of other downstream companies may not. Second, if a target supply chain is simple and global information sharing is possible, alternative strategies such as global planning can be used. But today, many supply chains have complex structures and they can be dynamically organized in the form of the virtual supply chain. Thus, the approach of MADC is more meaningful in supply chains with complexity and uncertainty.

Several further research issues remain. First, we plan to apply MADC to different models of supply chains with different assumptions and alternative measures of performance. Second, it is needed to analyze the performance of MADC with various demand patterns in markets. Third, it will be interesting to experiment with dynamic supply chain structure where the topology and parameters of a supply chain can change over time.

References

1. Yu, Z., Yan, H., Cheng, T.C.E.: Benefits of information sharing with supply chain partnerships. Industrial Management & Data Systems 101(3) (2001) 114–119
2. Verwijmeren, M., Vlist, P., Donselaar, K.: Networkd inventory management information systems: materializing supply chain management. International Journal of Physical Distribution & Logistics 26(6) (1996) 16–31.
3. Zhao, X., Xie, J., Zhang, W.J.: The impact of information sharing and ordering co-ordination on supply chain performance. Supply Chain Management: An international journal 7(1) (2002) 24–40.
4. Simatupang, T. M., Wright, A.C., Sridharan, R.: The knowledge of coordination for supply chain integration. Business Process Management 8(3) (2002) 289–308.
5. Lau, J.S.K., Huang, G.Q., Mak, K.L.: Web-based simulation portal for investigating impacts of sharing production information on supply chain dynamics from the perspective of inventory allocation. Integrated Manufacturing Systems 13(5) (2002) 345–358.
6. Berry, D., Naim., M.M.: Quantifying the relative improvements of redesign strategies in a P. C. supply chain. Production Economics 46–47 (1996) 181–196.
7. Boyson, S., Corsi, T., Verbraeck, A.: The e-supply chain portal: a core business model. Transportation Research Part E 39 (2003) 175–192.
8. Gavirneni, S., Kapuscinski, R., Tayur, S.: Value of Information in Capacitated Supply Chains. Management Science 45(1) (1999) 16–24.
9. Chen, F., Drezner, Z., Ryan, J.K., Simchi-Levi, D.: Quantifying the Bullwhip Effect in a Simple Supply Chain: The Impact of Forecasting Lead Times, and Information. Management Science 46(3) (2000) 436–443.
10. Cachon, G.P., Fisher, M.: Supply Chain Inventory Management and the Value of Information Sharing. Management Science 46(8) (2000) 1032–1048.
11. Raghunathan, S.:Information Sharing in a Supply Chain: A Note on its Value when Demand is Nonstationary. Management Science 47(4) (2001) 605–610.
12. Lee, H.L., So, K.C., Tang, C.S.: The Value of Information Sharing in a Two-Level Supply Chain. Management Science 46(5) (2000) 626–643.
13. Dejonckheere, J., Disney, S.M., Lambrecht, M.R., Towill, D.R.: Measuring and avoiding the bullwhip effect: A control theoretic approach. European Journal of Operational Research 147 (2002) 567–590.
14. Barbuceanu, M., Teigen, R., Fox, M.S.: Agent Based Design and Simulation of Supply Chain Systems. 6th Workshop on Enabling Technologies Infrastructure for Collaborative Enterprises (WET-ICE '97) (1997) 18–20.
15. Chen, Y., Peng, Y., Labrou, Y., Cost, S., Chu, B., Yao, J., Sun, R., Willhelm, B.: A negotiation-based Multi-agent system for supply chain management. Workshop on Agents for Electronic Commerce and Managing the Internet-Enabled Supply Chain, Seattle, WA, April (1999) 15–20.
16. Fox, M.S., Barbuceanu, M., Teigen, R.: Agent-Oriented Supply-Chain Management. Flexible Manufacturing Systems 12(2/3) (2000) 165–188.
17. Ito, T., Abadi, S. Agent-based material handling and inventory planning in warehouse. Journal of Intelligent Manufacturing 13 (2002) 201–201.
18. Min, J.U., Bjornsson, H.: Agent Based Supply Chain Management Automation. The Eighth International Conference on Computing in Civil and Building Engineering (ICCCBE-VIII) (2000)
19. Nissen, M.E.: Agent-based Supply Chain Dis-intermediation vs. Re-intermediation: Economic and Technological Perspectives. Intelligent Systems in Accounting, Finance, & Management 9 (2000) 237–256.
20. Nissen, M.E.: Agent-Based Supply Chain Integration. Information Technology & Management 2 (2001) 289–312.

21. Reis, J., Mamede, N., O'Neill, H.: Locally perceiving hard global constraints in multi-agent scheduling. Intelligent Manufacturing 12 (2001) 223–236.
22. Swaminathan, J. M.: Modeling supply chain dynamics: A Multiagent Approach. Decision Sciences 29(3) (1997) 607–632.
23. Verdicchio, M. Colombetti, M.: Commitments for Agent-Based Supply Chain Management. ACM SIGecom Exchanges 3(1) (2002) 13–23.
24. Walsh, W.E., Wellman, M.P.: Modeling Supply Chain Formation in Multiagent Systems. Agent Mediated Electronic Commerce Workshop (IJCAI-99) (1999) 94–101.
25. Yuan, Y., Liang, T. P., Zhang, J. J.: Using Agent Technology to Support Supply Chain Management: Potentials and Challenges. Michael G. DeGroote School of Business Working Paper Searies 453 (2001)
26. FIPA00025: FIPA Interaction Protocol Library Specification. Foundation for Intelligent Physical Agents (2001)
27. Pindyck, R.S., Rubinfeld, D.L.: Econometric Models and Economic Forecasts. 3rd edn. McGraw-Hill International (1991)

A Teamwork Protocol for Multi-agent System*

Qiu-Jian Sheng, Zhi-Kun Zhao, Shao-Hui Liu, and Zhong-Zhi Shi

Key Laboratory of Intelligent Information Processing,
Institute of Computing Technology, Chinese Academy of Sciences
P.O. Box 2704-28, Beijing 100080, China and
Graduate School of the Chinese Academy of Sciences
Beijing, China
shengqj@ics.ict.ac.cn

Abstract. Teamwork is an effective way of cooperative problem solving in dynamic and unpredictable application contexts. The concept of joint intention is the key of teamwork. How various speech acts can be used to form, maintain and dissolve the joint intention is a significant problem need to be investigated. The paper makes an in-depth analysis of the insufficiency of FIPA ACL with which agents to form the required joint intention in teamwork. We distinguish joint-request between delegation-request and extend the FIPA ACL by defining a new joint-request act whose properties are discussed. Based on the defined joint-request, we propose a teamwork protocol, associated with its formal semantic description and demonstrate its application as well. The teamwork protocol presents an interaction pattern that differs from those of existing elementary request protocol, contract-net protocol and auction protocols within FIPA Interaction Protocol Specifications. The proposed protocol can facilitate design of interaction modules in multi-agent teamwork.

Keywords: joint intention; ACL (Agent communication language); joint-request; delegation-request; teamwork protocol

1 Introduction

A multi-agent system (MAS) is usually built up from a group of autonomous agents designed independently. By a knowledge level communication between agents, they can cooperatively accomplish complex tasks that any individual agents fail to do. Compared to the traditional intelligent systems, e.g., expert systems, MAS represents a new computing paradigm. MAS computing can be divided into two levels, that is, individual behaviors and social behaviors. The social behaviors of agents are crucial for MAS to achieve social intelligence, e.g., Robocup. Communication between agents by ACL (agent communication language) is an important way to implement social interactions. ACL is an effective vehicle for agent to share knowledge, communicate mental states and coordinate their behaviors. Thus, it has been one of important researches of agent community [3][4][6][7][8][15] in recent years.

* Supported by the National High-Tech Project (Grant No.2001AA113121) of China National Natural Science Foundation of China (90104021), and Beijing Natural Science Foundation (4011003)

J. Lee and M. Barley (Eds.): PRIMA 2003, LNAI 2891, pp. 25–36, 2003.

Teamwork is an effective way of cooperative problem solving in dynamic and unpredictable application contexts [14][16][17][18]. Central to agent teamwork is that team members have a shared goal and shared mental states about current team activity in which agents are jointly engaging [2]. The formation and evolution of the joint mental states (e.g., joint intention) among member agents is a social interaction heavily driven by ACL communication.

FIPA ACL is a recently developed agent communication language [3], which has been increasingly adopt by more and more agent platforms [5] and expected to be a standard of agent communication. At present, FIPA ACL includes many speech acts, e.g., query, inform, etc. which support agent to participate in elementary social activities. However, how various FIPA speech acts can be used to form, maintain and dissolve joint intention(team) in order to support advanced social activity (i.e., teamwork) is a significant problem need to be investigated.

The paper investigates on the problem that how a group of FIPA agents can perform teamwork by talking FIPA ACL with each other. Our contribution is twofold: 1) we distinguish the conception of joint-request between delegation-request, analyze the insufficiency of FIPA ACL and extend it by introducing the joint-request act. 2) we design a teamwork protocol for multi-agent system. It describes the interaction pattern of cooperation in agent teamwork. The protocol is useful in that it can normalize the design of interaction module of team agents, structure the interaction process among agents to make sure that team has a consistent joint activity, thus, preventing agent team from running into chaos.

The remainder of this article is organized as follows. Section 2 gives a brief introduction of joint intention theory. Section 3 distinguishes the conception of joint-request between delegation-request, analyze the insufficiency of FIPA ACL and extend it by introducing the joint-request act. Based on the joint-request act, we further discuss the team formation and disbandment through communicating extended FIPA ACL acts. Section 4 describes the protocol in AUML and its formal semantics in modal logic. Section 5 discusses a domain application of teamwork protocol. Section 6 concludes the article.

2 Joint Intention

Joint intention theory [2] deals with agent's rational balance in performing joint actions within teamwork. It turned out that shared goal and shared mental states, i.e., joint commitment and joint intention of team are the key to achieve consistent joint action for team. The joint intention theory is expressed in a modal language and linear time temporal logic. Operator BEL and MB denote agent's believe and mutual believe among agents respectively. Goal and persistent goal are expressed in GOAL and PGOAL respectively. $\diamond$p means the proposition p will eventually be true, and $\Box$p means the p will always be true. HAPPENS and HAPPENED says action evens will happen next and has happened, Doing and Done mean action evens are ongoing and have been done respectively. UNTILL is defined thorough HAPPENS and Done, see [1] for details.

Definition 2.1 Weak Goal
WG(x, y, p, q) ≡
[¬ BEL(x, p) ∧GOAL(x, ◇p)]∨[BEL(x, p) ∧ GOAL(x, ◇MB(x, y, p))] ∨
[BEL(x, □¬ p)∧GOAL(x, ◇MB(x, y, □¬ p))] ∨
[BEL(x, ¬ q)∧GOAL(x, ◇MB(x, y, ¬ q))]

WG says that agent x believes p is false currently and x want to achieve p, or x believes that p is finished (i.e., p is true or unachievable, or irrelevant) and want to make that mutually believed.

Definition 2.2 Joint Persistent Goal
JPG(x, y, p, q) ≡
MB(x, y, ¬ p)∧MB(x, y, GOAL(x, ◇p)∧GOAL(y, ◇p)) ∧ (UNTIL [MB(x, y, p) ∨ MB(x, y, □¬ p) ∨MB(x, y, ¬ q)], MB(x, y, WG(x, y, p) ∧WG(y, x, p)))

Two agents x and y have a joint persistent goal p with respect to q when precisely the following conditions hold: there is a mutual belief that p is not currently true, it is a mutual goal to bring about p, and p will remain a weak mutual goal until there is a mutual belief that p is either true, or will never be true, or the relative condition q is no longer true.

Definition 2.3 Persistent Weak Achievement Goal
PWAG(x, y, p, q) ≡
[¬ BEL(x, p)∧PGOAL(x, p)] ∨[BEL(x, p) ∧PGOAL(x, ◇MB(x, y, p))] ∨
[BEL(x, □¬ p) ∧PGOAL(x, MB(x, y, □¬ p))]∨ [BEL(x, ¬ q) ∧PGOAL(x, ◇MB(x, y, ¬ q))]

A persistent weak achievement goal (PWAG) is the building block for establishing JPG. The term PWAG states that an agent x has a persistent weak achievement goal with respect to another agent y to achieve p relative to q. PWAG is a longer lasting version of the weak goal used in the definition of JPG. Mutual belief in each other's PWAG towards the other is sufficient to establish a JPG between two agents provided that the PWAGs are interlocking i.e. if one PWAG is relative to the other.

Definition 2.4 Joint Intention
JI(x, y, a, q) ≡
JPG(x, y, Done(x, y, [UNTIL (Done(x, y, a), MB(x, y, Doing(x, y, a)))] ?: a),q)

A joint intention of x and y means they have a joint persistent goal to p relative to q, meanwhile, they mutual believe they are all doing it.

3 Joint Request

In this section, we take the joint intention theory as a tool to analysis the semantic requirement on ACL in order to support teamwork, and highlight the distinction in notion between joint request and delegating request. In terms of it, the insufficiency of FIPA ACL is pointed out then a new joint request act is defined. Also, we will discuss the process of team formation and disbandment.

3.1 Joint Request versus Delegating Request

We take a simple example to help our analysis. Assume there are two agent, x and y with a goal to lift a table with a motivation to get a book from a shelf, in the case, p is to "lift a table", relative condition q is "to get book from a shelf with the table". Moreover, we suppose that either x or y has too limited capability to lift the table individually. Thus, they need to work as teammate with each other to achieve p collectively.

As an initiator agent x requests agent y jointly work on p. According to the joint intention theory, they must make a jointly commitment towards p. But how can they get to the required mental state? Suppose that agent y agree to help with x once she receive request from agent x, thus agent y reply with agreement message to agent x. In principle, a team comes to existence at the moment when agent x receives an agreement reply from agent y and they still desire to achieve p. That is, JPG(x,y,p,q) holds. Specifically, the following mental states should be true: agent x and y mutually believe that p is currently false, and; agent x and y mutually want to achieve p, and; x and y believe that they all keep a weak goal towards p unless they mutually believe that p is finished (i.e., p is true or unachievable, or irrelevant).

It is not difficult to know that with a request act agent x have an object to joint agent y to accomplish p collectively, but not desire to delegate agent y to make p done individually. It is important to distinguish these two cases. For the purpose, we precisely call the former request joint-request and the latter delegation-request notably. Essentially, they need to convey different semantics and mental constraints on sender and receiver of request act.

The joint-request need to convey three points in order to form a team, with agent y's agreement, between agent x and y. Firstly, to let agent y know p is currently false and intend to make p true; Secondly, to ask agent y to take p as a PWAG with respect to request of agent x, thus, agent y commit to synchronize the state of p with agent x; Thirdly, agent x commit to synchronize the state of p with agent y once he privately believe, namely, agent x himself take p as a PWAG with respect to agent y.

Essentially, the first point ensures that agent x and y have a shared goal and the rest two points make sure that agent x and y have shared mental states about p during cooperation. It is the latter points that constraint the team member not allowed to freely leave from the team.

In contrast to the joint-request, the delegation-request needs only convey the first point of the joint-request discussed above. In other word, delegation-request intend to tell agent y to do p singly without cooperation of agent x. Thus, it is obvious that even agent y reply with agreement message to agent x, there exists no JPG(x,y,p,q) to wards p yet. Therefore, delegation-request is insufficient to support social interaction in teamwork.

Based on the detailed analysis above, let's investigate request act of FIPA ACL[3].

request≡
<x, request (y, a)>
FP: FP (a) [i\y] ∧ BEL(x, Agent (y, a)) ∧ BEL(x, ¬PGOAL(y, Done (a))
RE: Done (a),

where, FP(a) [x\y] denotes the part of the FPs of action a which are mental attitudes of agent x. It is obvious to know that it agent y is only expected to intend to make

action a done alone, Besides, there is no additional constraints, neither on agent x nor on agent y, that require them to take a p ,i.e., Done(a) as a PWAG. Thus, FIPA request is essentially a delegation-request. It conveys too little semantics to form the required JPG for consistent teamwork.

3.2 Joint Request

3.2.1 Definition of Joint-Request

In this section, we give a formal definition of joint-request compatible with FIPA ACL.

Definition 3.1 joint-request (x, y, a, q)≡

request(x, y, a);inform(x, y, ϕ1);request-whenever(x, y, ϕ2, ϕ3), where

ϕ1= PGOAL(x, p)$\wedge$[[BEL(x, p)$\wedge\neg$ BEL(x, BEL(y, p)) $\Rightarrow$

PGOAL(x ,MB(x, y, p))] $\vee$

[BEL(x, $\Box\neg$ p) $\wedge\neg$ BEL(x, BEL(y, $\Box\neg$ p)) $\Rightarrow$ PGOAL(x ,MB(x, y, $\Box\neg$ p))] $\vee$

[BEL(x, $\neg$ q) $\wedge\neg$ BEL(x, BEL(y, $\neg$ q)) $\Rightarrow$ PGOAL(x ,MB(x, y, $\neg$ q))]]

ϕ2= inform(y, x, p)|inform(y, x, $\Box\neg$ p)|inform(y, x, $\neg$ q)

ϕ3= BEL(y, ϕ1)$\wedge$[[BEL(y, p)$\wedge\neg$ BEL(y, BEL(x, p))] $\vee$

[BEL(y, $\Box\neg$ p) $\wedge\neg$ BEL(y, BEL(x, $\Box\neg$ p))] $\vee$

[BEL(y, $\neg$ q) $\wedge\neg$ BEL(y, BEL(x, $\neg$ q))]],

where, p=Done(a), "|" denotes non-deterministic choice. Joint-request is defined based on three FIPA ACL acts include request, information and request-whenever [3]. It is a sequential composition of them. Intuitionally, it means that agent x requests agent y to perform joint action a with x, and inform y that he will synchronize the state of p with agent y, at the same time, agent y is also informed to synchronize the state of p with agent x during teamwork..

The feasibility preconditions (FP) and the rational effect (RE) can be derived from request, inform, request-whenever. Namely,

FP(joint-request (x, y, a, q)) =

FP(request(x, y, a)) $\wedge$ FP(inform(x, y, ϕ1)) $\wedge$FP(request-whenever(x, y, ϕ2, ϕ3)).

RE(joint-request (x, y, a, q))=

RE (request(x, y, a)) $\wedge$RE (inform(x, y, ϕ1)) $\wedge$RE (request-whenever(x, y, ϕ2, ϕ3)).

3.2.2 Team Formation and Disbandment

In this section, we mainly discuss the process of team formation and disbandment which are driven by ACL communication between agents. In order to focus on the semantic aspects of speech acts, we assume that agents are sincere and that communication is reliable.

Agent x joint-request agent y to do action a, implies that agent x has a persistent weak goal towards action a.

Theorem 3.1
(HAPPENED(joint-request(x, y, a, q))) ⇒ PWAG(x , y , Done(a), q).

Proof: According to the definition of joint-request given in section 3.2.1, agent x do not believe that action a has been finished, i.e., ¬ BEL(x, Done (a)).Moreover, agent x holds a persistent goal towards action a, i.e., PGOAL(x, Done (a)) , and promise to make state of goal(doing action a) mutual belief. i.e.,

[BEL(x, p)∧┐BEL(x, BEL(y, p)) ⇒ PGOAL(x ,MB(x, y, p))] ∨
[BEL(x, □┐p) ∧┐BEL(x, BEL(y, □┐p)) ⇒ PGOAL(x ,MB(x, y, □┐p))] ∨
[BEL(x, ┐q) ∧┐BEL(x, BEL(y, ┐q)) ⇒ PGOAL(x ,MB(x, y, ┐q))]],

where p=Done(a). Thus, PWAG(x ,y , Done(a), q) holds. ■

If agent x and y mutually know that joint-request was followed by an agreement reply, then there exists a joint persistent goal towards action a, that means a team is formed.

Theorem 3.2
╞MK(x, y, HAPPENED (joint-request (x, y, a, q);
agree(x, y, joint-request, true))) ⇒ JPG(x, y, Done (a), PWAG(x, y, Done(a),q)),
where MK(x,y,p)≡p∧MB(x,y,p).

Proof: From the definition of JPG, to prove JPG(x, y, Done (a), PWAG(x, y, Done(a),q)) exists, we need to show the following conditions are true. Firstly agent x, y mutually believe that action a has not been done; Secondly, they mutually believe that they all want to make action a done eventually; Thirdly, they must have a weak goal to do action a with respect to agent x's joint-request, and they must mutually know that this condition will holds for each of them until they mutually believe action a has been done, or unachievable or irrelevant or joint-request has been canceled.

The first and second conditions are obviously true. To prove the third condition, we must ensure that the state of goal must become a mutual believe between agent x and agent y. From the definition of joint-request given in section 3.2.1, inform and request-whenever acts add the required mental constraints on agent x and agent y, which make the third condition true. ■

If joint-request is followed by a refuse act then there exists no team.

Theorem 3.3
╞ (HAPPENED (joint-request(x, y, a, q));refuse (y, x, joint-request(x, y, a, q))) ⇒
¬Done(a) ∧ ¬JPG(x, y, Done (a),q)

Proof: It is easy to know, the proof is omitted. ■

Once a JPG has been achieved, we require a method to disband the team.

Theorem 3.4
╞MK(x, y, HAPPENED(joint-request(x, y, a,q);agree(y,x,joint-request(x, y, a, q),T);a;
inform(y, x,(Done (a)))⇒ [¬ PGOAL(x, y, Done (a), q)∧ ¬ PGOAL(y, x, Done (a),q)] ,
where T denotes true.

Proof: From theorem 3.2, we know that JPG holds after agree act performed by agent y. Inform(y, x,(Done (a))) implies agent y believed action a has been done, i.e., BEL(y, Done(a)). At this moment, agent y drops the corresponding persistent goal, i.e., ¬ PGOAL(y, x, Done (a), q). Upon receiving Inform(y, x,(Done (a))) from agent y, agent x believe that Done(a) is true, thereafter, he drops his persistent goal, i.e., ¬ PGOAL(x, y, Done (a), q). ■

To disband the team completely, they must establish a mutual belief that joint action a has been done. It can be achieved by agent x to confirm that he also believe action a has been done.

Theorem 3.5

⊨MK(x, y, HAPPENED(joint-request(x, y, a, q); agree(y, x, joint-request(x, y, a, q),T);a;inform(y, x,(Done (a)));confirm(x, y, Done (a)))

⇒ MB(x, y, Done (a)) ∧¬ JPG(x, y, Done (a),q),

where T denotes true.

Proof: Agent y's inform that Done(a) enables agent x to believe that action a has been done. Thereafter, agent x's confirmation lead to a mutual belief that Done(a), i.e**.,** MB(x, y, Done (a)).Thus, the joint persistent goal of team can be discharged, i.e., ¬ JPG(x, y, Done (a),q). ■

To sum up, we extended the FIPA ACL by introducing a joint-request act that was formally defined in section 3.1. Then we showed the capability of the extended FIPA ACL to support the formation and disbandment of team. Although it is proved that the extended FIPA ACL, in theory, can be used to support consistent teamwork completely, it is not necessary that a group of heterogeneous agents can reach a consistent teamwork in practice. In practical developments, one of the most time-consuming and difficult activities is to ensure the consistency of individual behaviors. Agent interaction protocol is an effective way to realize the goal. For the purpose, we are to design a teamwork protocol for multi-agent systems.

4 Teamwork Protocol

It is one of big challenges in agent community to enable heterogeneous agents to work instinctively as a team in dynamic and unpredictable application contexts. Due to autonomy of agents and complexity of environment where agent lives, multi-agent systems are easily run into chaos. Interaction protocol is an effective and efficient way to balance the autonomy of individual agents and the order of systems as a whole. It can be used not only to structure the interaction of agents but also to facilitate design of interaction modules in multi-agent systems. In recent years, agent interaction protocol has attracted more and more attentions in agent community [9][10][11][12]. Currently, FIPA ACL includes eleven agent interaction protocols [10]. They are designed with various purposes and reflect different interaction patterns. For example, FIPA request interaction protocol is used to describe the interaction between task delegating agent and delegated agent. FIPA contract-net protocols mainly are used to optimize the allocation of tasks among a group of agents. FIPA auction protocols specify the negotiation between seller agent and buyer agent.

In contrast, teamwork reflects a new interactive pattern that cannot be well specified by the existing protocols due to their different application purposes and different semantics they convey. The characteristic of the teamwork is the shared goal and shared mental states among interactive agents. To specify the interactive pattern of the teamwork, we design a teamwork protocol which can be used to normalize the interaction between agents in teamwork, thus, ensure that team agents works in a consistent way.

4.1 Protocol Description

The teamwork protocol is depicted in AUML [9] as Fig. 1. The Initiator requests other agents to work as team mates with each other by issuing a joint-request that specifies the goal to be achieved. Participant receiving the joint-request can reply with three kinds of communicative acts accordingly before deadline. Firstly, the Participant may not understand the content of message, then he replies a not-understand message. Secondly, the Participant may refuse to achieve the requested goal for some reasons, this case, a refuse message would be sent. Thirdly, if the Participant agrees to cooperate for requested goal, then he sent a agree message to the Initiator. In the former two cases, the protocol will be terminated. In the third case, the Initiator and the Participant make a joint commitment to perform the joint action, i.e., a JPG or team is established. During the process of accomplishing the shared goal jointly, any team member has obligation, due to their WGs, to communicate (inform/confirm) the state (joint action a has been done, or will never be achieved, or is irrelevant) of the goal in order to make it mutually believed among team.

Additionally, at any point of interaction, the Initiator may drop his joint-request for some reasons by issuing a cancel message.

4.2 Formal Semantics

In Figure 1, we intuitionally described the interaction of teamwork. In this section, we give the formal description in logic, showed in Fig. 2. S*i* denotes the (joint) mental states, e*i* denotes communicative action events, which drive the evolution and transition of states. s1 is the initiation state. s4, s5, s6 denote the goal of team is achieved, unachieved and irrelevant respectively.

S1: $\neg$Done(a) $\wedge\neg$JPG(x, y, Done(a),PWAG(x, y, Done(a),q) $\wedge$q)

S2: GOAL(x, $\Diamond\psi$) $\wedge$MB(x, y, PWAG(x, y, ψ, q))
ψ=Done(a) $\wedge$[PWAG(y, x, Done(a),PWAG(x, y, Done(a),q) $\wedge$q)]

S3: JPG(x, y, Done(a), PWAG(x, y, Done(a),q) $\wedge$q)

S4: Done(a) $\wedge$ MB(x, y, Done(a))

S5: $\neg$Done(a) $\wedge$
[MB(x, y, $\Box\neg$ PWAG(y, x, Done (a), PWAG(x, y, Done(a), q) $\wedge$q))
$\vee$MB(x, y, $\neg$ PWAG(x, y, Done (a), q))]

S6: $\neg$Done(a) $\wedge$[MB(x, y, $\Box\neg$ Done(a))$\vee$ MB(x, y, $\neg$ q)$\vee$MB(x, y, $\neg$ PWAG(x, y, Done (a), q))]

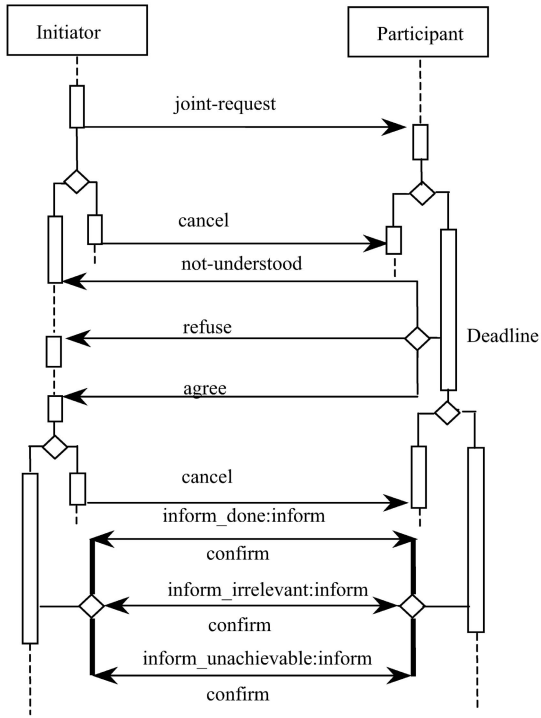


Fig. 1. The teamwork protocol in AUML

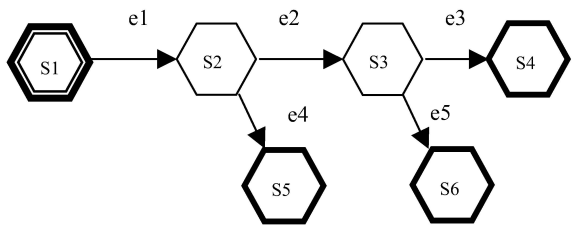


Fig. 2. Formal semantics of the teamwork protocol

The transitions of states are driven by the following communicative action events.

e1= joint-request(x, y, a, q)
e2=agree(y, x, joint-request ,T)
e3=inform(y, x, Done (a)); confirm(x, y, Done (a))
e4=refuse(y, x, joint-request)|cancel(x, y, joint-request)|deadline?
e5=e6|e7|e8|e9, where
e6=inform(y, x, ┐q); confirm (x, y, ┐q))
e7=inform(y, x, □┐Done (a)); confirm(x, y, □┐Done (a))
e8=inform(x, y, ┐q); confirm(y, x, ┐q)
e9=inform (x, y, □┐Done (a)); confirm(y, x, □┐Done (a))

5 Application

Teamwork is appropriate for problem solving in dynamic and unpredictable contexts. The teamwork protocol can ensure team agents have consistent teamwork in complex environments. Electricity transport network is a typically complex environment where various unpredictable faults often occur due to kinds of factors from the natural and human being. Therefore it is significant to diagnose the faults and restore the network quickly.

In this section we simply introduce an application scenario of the teamwork protocol. The scenario mainly involves three agents, including Monitor, Detector and Identifier. They cooperate with each other in order to diagnose the accurate faults. The Monitor is responsive for monitoring the electricity transport network (ETN). It would request Detector and Identifier to jointly diagnose the potential faults when it finds abnormal signals in ETN concerned. Detector first roughly calculates some possible faults quickly, according to the information of network topology provided by the Monitor. Then refine the approximate result to get accurate fault with the information of the black out area in current ETN, which can be worked out by the Identifier. The diagnosed fault information can be used to recover the ETN.

Because of the inherent dynamics of the ETN there may present some inconsistent cases during the process of fault diagnosis. For example, we assume that Monitor discover an abnormal signal at time T1 and send a joint-request message and some related information (e.g., topology information about the current ETN) to Detector and Identifier to diagnose the faults jointly. Thereafter, both Detector and Identifier start engaging in their own tasks. Here, we discuss two possible inconsistent cases at T2 (T2>T1). Firstly, Monitor realizes that the abnormal signal reported at T1 is transient which means the other two are attempting a nonsexist fault. This case, Monitor should send a cancel message immediately to them in order to avoid meaningless calculation which may potentially delay real fault diagnosis; Secondly, at T2, some further faults may occur, meaning the other two are calculating on the invalid information since network topology have changed due to the recent faults. In this case Monitor must immediately inform them that the team's goal at T1 is irrelevant due to the changes of the network topology. In order to diagnose the current faults correctly, Monitor would send the current status of the network to Detector and Identifier again. Figure 3 partially depicts the interaction of this case. Here, joint action a =”diagnose the accurate faults at T1” which is the common goal of team. The motivation q=”restore the failed ETN”. m,i,d,r respectively denote Monitor, Identifier, Detector and Restoring-agent which is responsive for restoring the failed ETN. The character “c” in message inform (i,d,c) denotes the content of back out areas information which is calculate by Identifier. Lastly, Detector send the accurate faults information to the Restoring-agent with inform (d,r,fault_info) message. The role of teamwork protocol followed by team members is to provide basis for determining which information should be shared and how agents should act when they receive. It can improve the overall consistency and robustness of teamwork by intelligently sharing information.

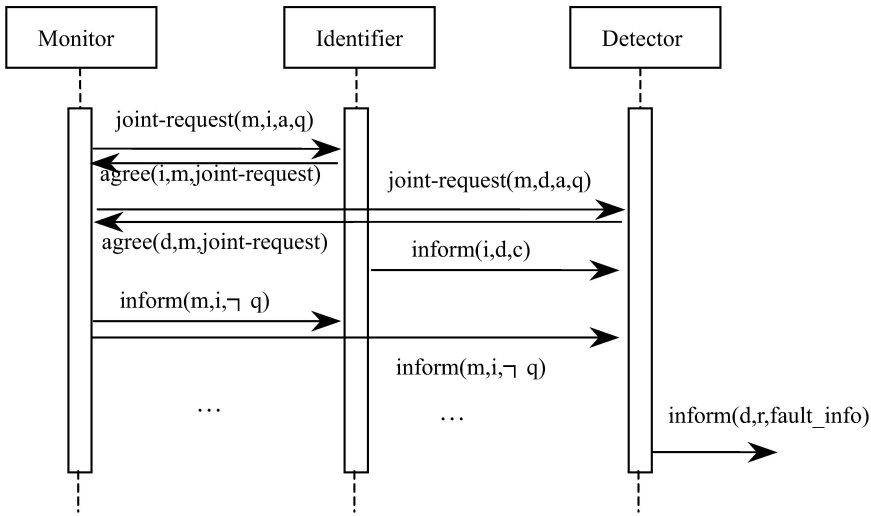


Fig. 3. Joint fault diagnosis

6 Conclusion

Teamwork is an effective way of cooperative problem solving, it has been playing important role in dynamic and unpredictable application contexts. We investigated on the problem that how a group of agents can perform teamwork by communicating FIPA ACL with each other. In the paper we distinguished the conception of joint-request between delegation-request then analyzed the insufficiency of FIPA ACL and formally defined a new speech act named joint-request act to extend it compatibly. Moreover, we formally discussed the process of formation and disbandment of team. Based on the newly defined joint-request act we proposed a teamwork protocol to facilitate the design of interaction modules in multi-agent teamwork. Then the semantics and application of the protocol were discussed. The teamwork protocol presents an interaction pattern that differs from those of existing elementary request protocol, contract-net protocol and auction protocols within the existing FIPA ACL.

Acknowledgements. The authors thank the anonymous referrers for their helpful comments on the earlier version of the paper.

References

1. Levesque, H. J., Cohen, P. R., and Nunes, J. H. T. On acting together, Proceedings of the Annual Meeting of the American Association for Artificial Intelligence, AAAI-90
2. Cohen, P. R., Levesque. Teamwork, Special Issues on Cognitive Science and Artificial Intelligence 1991,25(4):488–512.

3. FIPA (2002). Agent Communication Language. FIPA 2002 Specification. Foundation for Intelligent Phisical Agents, http://www.fipa.org, 2002
4. Smith, I. A. and Cohen, P. R. Toward a semantics for an agent communications language based on speech-acts, Proceedings of the Annual Meeting of the American Association for Artificial Intelligence (AAAI-96).
5. Agentcities Agent Network : Overview, http://www.agentcities.org/Network/index.jsp
6. B.Chaib-draa, F.Dignum, Trends in Agent Communication Language, Computational Intelligence, Volum18, Nomber2, 2002.
7. Tim Finin , Yanis Labrou , James Mayfield, KQML as an agent communication language, Software agents, MIT Press, Cambridge, MA, 1997.
8. M. P. Singh. Agent communication languages: Rethinking the principles. IEEE Computer, 31(12):40–47, 1998
9. Bauer B., Müller J.P., Odell J.R.E, Agent UML: A Formalism for Specifying Multiagent Interaction, In Paolo Ciancarini and Michael Wooldridge (eds) Agent-Oriented Software Engineering (Berlin 2001), Springer, 91–103.
10. Interaction Protocol Specifications, http://www.fipa.org/repository/ips.php3
11. S. Bussmann, N.R. Jennings and M. Wooldridge (2002), "Re-use of interaction protocols for decision-oriented applications" Proc. 3rd Int Workshop on Agent-Oriented Software Engineering, Bologna, Italy.
12. M.-P. Huget, J.-L. Koning, Interaction protocol engineering, in M.-P. Huget, editor, Communication in MAS: background, current trends and future, Springer-Verlag, 2003.
13. Yi. Li, Cun yi. Shi, A semantic Description of An Agent Communication Language, Chinese journal of computer research and development, Vol. 39,No. 6, 2002,pp. 696–700.
14. Jing.Li, Zhao-Qian Chen, *et al.,* Suvery of Multiple Agents Teamwork, Chinese journal of computer research and development, Vol. 39,No. 6, 2002, pp. 421–429.
15. Zhongzhi Shi, Intelligent Agent and its Application, Science Press, 2001 (in Chinese).
16. Frank Dignum, Rosaria Conte: Intentional Agents and Goal Formation. ATAL 1997: pp. 231–243
17. M. Wooldridge and N.R. Jennings. The Cooperative Problem Solving Process. In Journal of Logic and Computation, 9(4), pp. 563–592, 1999
18. Tambe, M., and Zhang, W., Towards flexible teamwork in persistent teams: extended report Journal of Autonomous Agents and Multi-agent Systems, special issue on "Best of ICMAS 98", 2000

Extraction of Implicit Resource Relationships in Multi-agent Systems

Toshiharu Sugawara, Satoshi Kurihara, and Osamu Akashi

NTT Network Innovations Laboratories
Tokyo 180-8585 Japan
{sugawara,kurihara,akashi}@t.ecl.net

Abstract. This paper discusses the storage and analysis of past hierarchical-planning results in order to identify implicit costs and resource relationships between activities in multi-agent contexts. We have previously proposed a plan-reuse framework in which plans are stored as templates after use and then reused to speed up planning activity in multi-agent systems. In this paper, we propose the mechanizm for learning, from templates that consist of used plans and data recorded during planning and execution, implicit relationships concerning resource usage by multiple agents. Here, implicit indicates that the relationships exist in the environments where agents are deployed but are not described in the domain models the agents have. The plan-reuse framework also provides guidance on which data the planner and executor should record and on when the learned rules should be applied. Finally, some examples show how this learning enables the creation of more appropriate solutions by agents.

1 Introduction

In the field of multi-agent systems (MAS), coordination is an important research area for realizing effective and efficient activity of agents. The key issues addressed in coordination are conflict detection and resolution. Conflicts may originate from the use of scarce resources or difference of intention among agents. The detection and resolution of conflicts is usually heavily reliant on the local knowledge of the agents (models of domains, tasks, resources, and other agents). Anticipating possible conflicts is, however, often difficult because of the uncertainty in information about the other agents' states. Furthermore, the models that guide the agents may lack representations of certain necessary features that are almost impossible to describe completely at design time and conflicts will occur because of the individual environments where the agents are deployed. This paper focuses on some unexpected performance introduced by undescribed (and thus unexpected) factors in relation to resources and agent interaction.

These undescribed relationships between agents often become evident when the observed data on duration, costs, and resource usage in task execution are quite different from the expected data. For example, if a robot has to move up a slope but this slope has not been represented in the domain model, an unexpected extra cost will be incurred. As another example, suppose that a battery shared by two robots is described as having a capacity of 36 V and 200 mA in the model, but if it is actually used at full current, its voltage slightly falls. In this case, the robot may fail to accomplish the task.

J. Lee and M. Barley (Eds.): PRIMA 2003, LNAI 2891, pp. 37–49, 2003.

The purpose of this paper is to propose a method by which an agent identifies those situations where some elements of the cost or duration of task execution are not encompassed by the known task and resource structures and learns these missing structures from experience. We call the possible missing structures *implicit relationships*. The issues we need to address in devising ways to learn implicit relationships are how to: collect those data by each for the same problem-solving situation that indicates the existence of unknown interference, extract the implicit relationships (as rules), and identify situations where such relationships exist (so the agent can take them into account).

We have previously proposed a plan-reuse framework in hierarchical planning (or HTN [3,4,8]) to speed up planning[9]. The used and executed plans are stored as *(plan) templates* with information related to detected and resolved conflicts, execution time, resources used, and so on. The templates are collections of past plans and are reused in the same or similar situations, so we propose a method for extracting resource-related implicit costs and relationships among tasks from templates that record past experience. The plan-reuse framework can also guide agents in deciding when the recorded data should be retained and the extracted relationships should be taken into account. Thus we propose to incoprate this learning method into the plan-reuse framework.

This paper is organized as follows: we start by briefly explaining hierarchical planning and plan reuse and show an example problem. We then discuss some requirements that the proposed form of learning imposes on the planner and executor. After that, the data structure of the plan templates and how they are generated are shown. Next, we explain the learning method used to extract the implicit relationships. Finally, we look at the example scenarios where this learning method is applied to an example problem.

2 Background and the Basic Example

2.1 Hierarchical Planning and Plan Reuse

In hierarchical planning, the initial states and goals are initially described in a model at the highest level of abstraction and a number of task sequences are generated to achieve these goals. One of these sequences is selected according to some planning strategy. Then the planner refines each task in the sequence into the sub-task sequences for a less abstract model that comes closer to actually achieving the task (these sequences are called *refinements* of the task) and selects one of these. This process is iterated until all tasks in the sequences have been refined to primitive tasks that correspond to directly executable procedures. We assume the popular STRIPS-style plan representation, in which all tasks have pre- and post-conditions that are referred to during generation of the plan. In this paper, a (hierarchical) plan is a collection of the task sequences in all layers of the abstraction hierarchy. The sequences in a certain layer are expressed in the model corresponding to this layer (see Figs. 1 and 2).

Plan reuse is a planning framework in which used plans are stored as (plan) templates and reused on similar problems in future. Plan reuse is thus applicable to problem domains where the environment changes infrequently. Plan reuse in MAS can speed up planning by avoiding the reiteration of planning activities that include the same communications, detection-makings and resolutions of the same conflicting/redundant

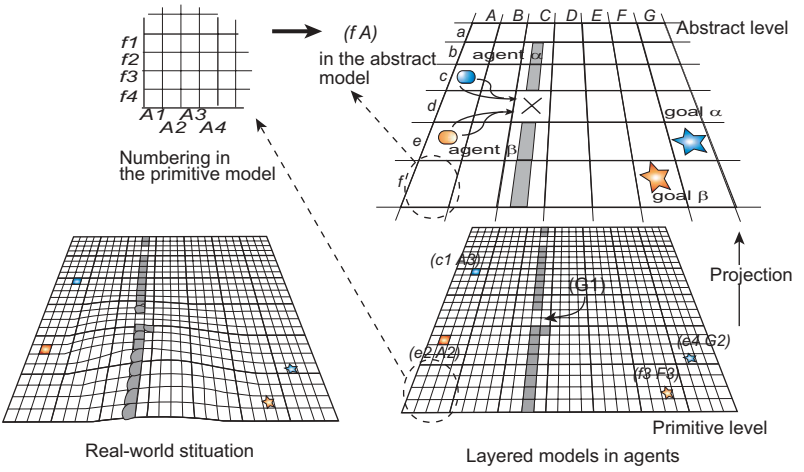


Fig. 1. Model held by agents and a realworld situation

activities. Plan reuse has been proposed for the single-agent case [6] and for MAS [9]. In both cases, the system stores generated plans as templates; when it encounters a problem, it retrieves a plan template that was used for an identical or similar problem (given problems are expressed in the abstract layer and if two problems are identical in that layer, we assume that they are similar to each other) and modifies the template to be applicable to the current problem. The details of plan reuse, such as the structures of stored plans and use of past plans as bases for planning are described in [9].

2.2 Example Problem

Our example is the route exploration problem for robots (agents) shown in Fig. 1, where a 3-layer model is used in the hierarchical planner. In the 0th- and 1st-layer models, which are identical and both called abstract models, the details are omitted and simplified, as in a large-scale map, but it is easy to create global plans. The lower-layer (or primitive) models express the details of the route but have a larger search space.

As the initial (default) conditions for planning, an agent moving to the next square in the primitive model takes one tick and consumes one unit of power. This is described in all agents' models. However, the real-world setting is a little more complicated: (1) there is a hill around the gap G1, (2) moving up/down to a next square via this hill takes each agent two ticks and twice as much as power, and (3) recognizing the walls, obstacles and other agents on a one-square wide path to avoid collisions requires additional three ticks and 50% more power than the normal case. the gap (G1) in Fig. 1 is two squares wide so the additional time and power in (3) will not usually be necessary, but if another agent is trying to get through the gap simultaneously, both agents must travel along the one-square-wide path. Extra cost and time are thus incurred. The actual cost and time increases in (1) to (3) are not described in the models, and are thus implicit.

Suppose that two agents intend to pass through the gap. They start by generating plans at the abstract level. They can then infer the possibility of collision (but cannot

conclude that a conflict will actually occur). Next, they proceed to the primitive layer to create their primitive plans. After some communications between the agents, they find two disjoint routes pass through the gap so they can pass through simultaneously.

However, in the actual execution of the plans, they will record higher costs than expected ones when they pass through the gap. The objective of this paper is to give the agents a way of drawing on experience in solving similar problems by using past plans and experience in the form of the recorded data on costs and resource usage. This is used to give the agents a basis for finding negative relations (in this case, the *hindrance* relation) between the tasks of the two agents.

2.3 Situation-Specific Learning

The rules learned in our method are agent-, environment-, and situation-specific. The plan-reuse framework provides the important features for this type of learning. Firstly, the agent needs to be able to compile and embed monitored data into a segment of plan template which corresponds to the executed plan. All data embedded at a given position on the template are recorded in the same or similar situations. Secondly, this makes additional situation-specific monitoring possible . As is later described, some data are costly to collect. Agents can collect such data only when they are necessary. Finally, agents have to be able to identify when the extracted relationships (rules) should be taken into account during planning. They should only apply a rule when the segment of the template from which it was extracted is to be reused.

3 Planner and Executor

Each agent has a planner and executor. For the proposed learning, it is assumed that both are able to record the following data on their activities.

(a) Generated and executed plans.
(b) Knowledge used during planning, such as:
  - anticipated resource requirements (electric power, tools, rooms, etc.),
  - anticipated execution times of tasks,
  - conflict relationships, and
  - conflict resolution methods.
(c) Observed execution records, such as:
  - execution time and used resources,
  - positions of other agents, and
  - time relations between local and non-local task execution.

Notice that while the data (a) and (b) for attachment to plans are usually detected on the basis of given local knowledge, the data (c) are obtained through actual execution.

For example, in our example problem domain, task *(f1 A1)* $\rightarrow$ *(f1 A2)* requires an empty square at *(f1 A2)*, one tick (anticipated time) and one unit of electric power. Hence these requirements are recorded.

Generated and executed plans that will be stored as templates can be classified into the following types:

- Plans that were completely executable:
  (p1) done in around the expected time with the expected usage of resources, and
  (p2) done, but in a longer time and/or with greater resource usage than was expected.
- Plans that were not completely executable, because:
  (p3) they led to an unexpected state, and
  (p4) the agent stopped executing the plan, due to a failure.

Agents identify a type of the plan by comparing data on what they can expect as calculated in planning, with data on the results of the plan's exectuion. Each agent then stores its plan and the related data according to its type.

Our aim is to avoid unexpected costs and resource usage (i.e., the first item in (c)) which are caused by certain interactions between agents (corresponding to the second and third items in (c)). Hereafter, the latter are called *relative data*, which are the basis of relationships that will be extracted, and the former are called *performance data*.

4 Recorded Plans, Resources, and Conflicts

This section discusses how the data monitored and recorded during planning and execution are expressed in templates. An example will be given in Section 4.5.

4.1 Descriptions of Plan Templates

Plans used in past are stored as (plan) templates in the agent's database. Details of the structure and construction of templates are omitted here, because this paper is only concerned with the learning of implicit relationships; we thus give only the overview that will be needed for the subsequent discussion.

A template is a collection of past plans generated for problems with identical expressions in the abstract model. After problem solving, the plan used is recorded in the planner. Each used plan consists of: task nodes (at all levels) and *annotations* attached to all nodes. A task node corresponds to a task in a certain layer. A non-primitive task node has a sequence of lower-level task nodes that expresses the refinement of the used plan. An example structure is illustrated in Fig. 2. A number of refinements had been generated to achieve each non-primitive task, but only one was actually executed.

An annotation is a description of recorded data that is retained by planner and executor (see (b) and (c) in the previous section). The annotation which corresponds to an item of performance data is a pair of an *attribute* and value, although relationships between tasks and between tasks and environment factors are denoted by logical formulae: some important annotations will be described in this section.

Finally, two types of node (*environment node* and *agent node*) are introduced to describe the environment and another agent that may interfere with the local agent.

4.2 Types of Resources

We introduce two pairs of resource properties: divisible/exclusive and consumable/non-consumable. The amount of a resource and the duration over which the resource is used are also important concepts in describing resources.

- **Divisibility:**
  - **Divisible:** This property means that a resource can be shared among agents. For example, 1 A of current can be divided into 500 mA for each of two robots.
  - **Exclusive:** This property means that a resource is used exclusively by a single particular agent and cannot be shared with other agents while it is in use. For example, it is impossible for two robots to equally share a battery charger.
- **Exhaustion:**
  - **Consumable:** A resource of this type is consumed by usage and may be used up if agents use large amounts of it or use it for a long time. For example, a battery will be used up unless it is recharged.
  - **Nonconsumable resource:** This type of resource does not run down or is non-expendable. On the local scale, for example, electric power from an outlet can be considered as being of this type.

Note that some resources impose additional conditions; for example, a solar battery is divisible and non-consumable but is not applicable to dark places.

The type of a resource is assumed to be given as knowledge on the resource. The planner can use the types to calculate certain resource-related conflicts. The annotation Use(tnode,rname,amount,[duration]) denotes the expected or monitored usage of a resource, where tnode is the task (with its executor agent), rname is the resource used, amount is the amount of the resource used, and duration is the time over which the resource was used in the task (if omitted, duration is identical to the expected duration).

4.3 Mismatches and Conflicts

We define a *mismatch* as an inconsistency between local tasks (that is, between the pre-conditions and post-conditions of tasks) and between a local task and the agent's environment. Here, local indicates the agent we are considering. A *conflict* is some interference between the task in the local agent and a task in another agent.

A mismatch is denoted as mismatch(t1,t2,rp,m), where t1 and t2 are tasks such that t1's post-conditions violate t2's pre-conditions and there is no other task between t1 and t2 that recovers t2's pre-conditions; m is the method which repairs this mismatch; and rp is the replaced part of the task sequence. t1 may be an environment node.

We distinguish resource-related conflicts from other types of negative conflicts between agents and classify five types of such conflicts.

- **Conflicts Caused by Resources:**
  - **Deficit relation:** We refer to this (hard) conflict as a disabling relation produced by local and/or non-local tasks accessing scarce or insufficient resources and is denoted by deficit(tnode,tnode',[resource]).
  - **Hindrance relations:** This is a soft resource conflict which does not make task execution impossible. If a task requires a scarce or insufficient resource, it may take longer to perform or its result may not be of good enough quality. This is denoted by hindrance(tnode,tnode',[resource]).

Table 1. Examples of methods of resolution

Method	Description
synchronization	Stop until another agent performing a task that requires a needed resource finishes the task and releases the resource. Wait for a primitive task or use of some resource by another agent until the task finishes or the agent releases the resource.
Waiting	Stop until other agents finish tasks that accomplish pre-conditions of the local task.
replacement	Replace tasks whose post-conditions do not affect tasks in other agents or whose pre-conditions are not affected by tasks in other agents
reordering	Reorder tasks to avoid negative relations.
insertion	Insert tasks whose post-conditions recover the pre-conditions of the task.
commission	Entrust the task to other agents. This form of resolution is preferable when, for example, a conflict can only be resolved by other agents or another agent can do the task at lower cost.

– **Conflicts Caused by Task Structures or Execution Orders:**
  Conflicts of intention: A conflict of this type is caused by different intentions among multiple agents so that joint tasks have to be postponed. This will usually only occur in collaboration; e.g. when an agent has been asked to cooperate in a task by another agent but does not intend to do so.
  Clobber relations: This hard conflict is a kind of disable relation and indicates the situation where one agent performing a task deprives another agent of the pre-conditions for one of its tasks. All resource-related conflicts are expressed by hindrance and deficit relations, so this relation only expresses other types of conflicts. For example, if doing a task requires that an agent moves to a distant place, it will be unable to collaborate with the other agents near its home.
  Delay relations: This is the kind of soft conflict where other agents' tasks lead to an increase in task duration for the local agent. For example, a local agent might intend to grasp a block and then move it, but another agent, during its preceding tasks, moved the block to the other side of the local agent. This does not disable the agent with respect to the task, but prolongs its execution.

Conflicts are usually resolved in one or more of the ways listed in Table 1. The system-defined resolution strategy determines which resolution method is applied.

The annotation for a conflict resolution is denoted by ConfRes(conf,m,rp,[cost]), meaning that the conflict indicated by the term conf was resolved by method m and the resulting part of the task sequence is pointed to by rp. The term cost, which is optional, indicates the expected cost if the resolved task sequence is followed.

4.4 Relativity, Cost, and Duration

Performance data, e.g. on amounts of resources used and times taken to execute individual tasks are stored as values of the corresponding attributes in each task node. Relative data are expressed as follows: time relations are those proposed by Allen[1], such as "overlap," "before," and "after." The relative locations of agents that we use in the example are "same," "left-hand side," "right-hand side," "neighbor" (= left- or

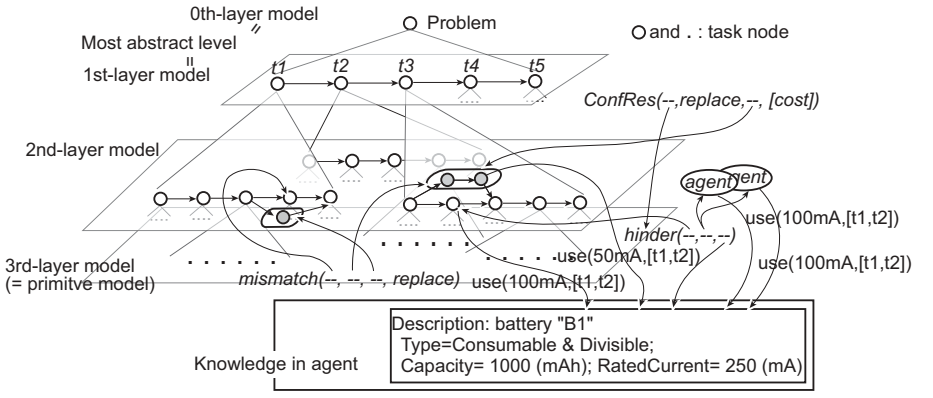


Fig. 2. Conceptual template structure

right-hand side), "front," and "back" (so other relative locations are not of interest in this case but other relative locations should also be expressed if they may affect an agent's performance). For example, two agents on squares in the primitive model of Fig. 1 have a neighbor relation if they are standing on adjoining squares and this relation may affect the level of power consumption and time needed in the way described in Section 2.2.

An anotations on the relative locations of agents and time relationships that affect resource usage takes the form of a pair of primitive tasks, with one on each agent. For example, neighbor(task1,task2) indicates that task task1 in one agent and task task2 in another agent were being executed at almost the same time and that the agents adjoined each other during the execution of these tasks; overlap(task3, task4)indicates that, at some stage, one agent was performing the task task3 while another was performing task4. Note that "taskX" contains information on its actor agent.

Performance and relative data recorded by the executor are attached to each primitive task. These data for a non-primitive task are easily computed by summation and/or concatenation in a bottom-up manner. Note that agents can anticipate some performance and relative data during planning, so both the anticipated and actually observed data are embedded in the templates.

As mentioned in Section 2.3, further cost is incurred in the monitoring and recording these data. This is especially true of the relative data, because the agents must communicate with each other to obtain these. When this cost is non-negligible, agents can get the data in two phases. In the first (normal) phase, the agents record the low-cost data and try to extract rules for implicit relationships from them. However, if agents cannot find a way to clearly identify the situation where the rules should be applied, the agents move to the second phase, in which other data to do with relativity are also recorded during execution. The plan-reuse framework provides effective control for determining when agents should enter the second phase. An agent enters the second phase only when it encounters an unexpected situation in executing a plan such that the learning procedure requires the collection of second-phase data for a new part of the templates.

4.5 Example of Templates

Annotations are generated from the recorded processes of the planner and the executor. An example is shown in Fig. 2. In this figure, the task in the (local) agent is hindered by certain tasks in the other two agents (because all three tasks draw on the same resource "B1," and require more than the rated current). This is expressed by a hindrance relation. The planner uses replace as the resolution method, replacing the task with other task sequences. The pre-condition of one of the new tasks is, however, violated by a local task in another task sequence (mismatch). So, by applying a method of repair (in this case "replace"), the blocking task is replaced by another task. None of the observed performance and relativity data are included in this figure.

5 Extraction of Implicit Relationships

5.1 Implicit Relationships

Let us recall the meanings of the hindrance and deficit relations as given in Section 4.3 and determine when these implicit relationships may exist.

hindrance: This conflict lowers performance levels due to the simultaneous use of resources or dodgy resources (tools), so that, relative to the results of planning, the agent takes longer to execute a task, consumes more resources, or occupies an exclusive resource for a longer period. In our example problem in Section 2.2, the relative locations of agents may affect their performance in movement (square occupancy is a kind of resource), especially in the case where they are in adjacent squares. So we can say that there is an implicit hindrance relation in these tasks.

deficit: The deficit conflict means that a task cannot be achieved because of a resource shortage (so that plan execution stops at this point). Other factors are the same as in the case of the hindrance relation. Of course, the planner generates plans to avoid this problem, but the agents will meet deficits that have not been described.

5.2 Learning Procedure

Step of Data Collection: The data required to identify implicit relationships are the executed plans with the associated records of planning and execution. The agents uses performance data (cost and task duration) in their data sets (defined below) in this and subsequent steps. These data are compared with the expected values. Retention of performacen data starts whenever unanticipated performance or resource usage is found in a data set. When unexpected value of an attribute is found, the value should be revised by new rules generated from data observed during plan execution. The part of the plan which contains this value and is also, of course, part of the template, is called the *interesting plan segment* (*IPS*). A *data set* is the set of retained data collected between one round of planning and execution. In our approach, plans are generated using templates (collections of past plans), so the new data set is attached to the corresponding IPS.

Step of Clustering: This step is invoked when the number of data sets collected for an IPS goes beyond a given threshold. In the previous step, the values of the attributes in

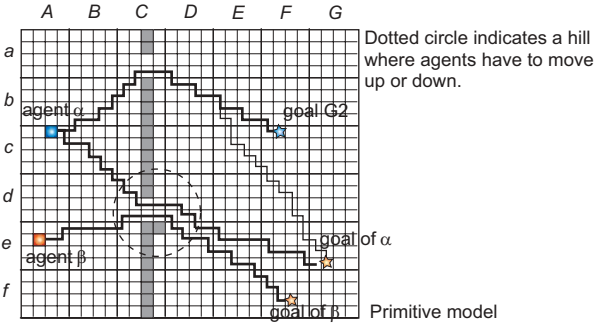


Fig. 3. Example problem in the route-exploration domain

the data sets for each primitive task in the IPS have been compared to those that are different from the expected values. The data sets are then clustered into groups based on the results of this comparison: (c1) the cluster whose data sets with expected values; and (c2) the cluster of data sets with unexpected values. The data sets in (c2) are clustered into smaller groups based on the values of the considered attributes.

Step of Identification of Common Features: This step involves reference to relative data. For each cluster of data sets, attributes that characterize particular situations are extracted by comparing attribute values. Such common attribute values are relative data. That is, they take such forms as time relations of resource usage, time relations of task execution, location of the local agent, and locational relationships (relative locations) between the local and other agents. All data sets in the cluster are then compared and any differences between the relationships and/or execution times are detected.

If, in the above process, it finds no difference between the data sets in the cluster that corresponds to the IPS being generated in response to the unexpected value, all of the agents enter the second phase of data monitoring. That is, the agents then begin to collect data sets for reuse in solving subsequent problems.

Hence the rules extracted, which reflect implicit relationship are attached to the IPS in its template. Then, when the segment is reused on a new problem in the future, the new rules are taken into account. (An empty (or almost empty) cluster (c1) suggests that the agents must always include the implicit relationships that have been identified.)

6 Example Scenarios

This section shows how rules required to cope with implicit relationships are imported in the proposed method of learning, using the example in Figs. 1 and 3. Each of two agents has its own problem: agents α and β have to find routes *(c1 A3)* $\rightarrow$ *(e4 G2)* and *(e2 A2)* $\rightarrow$ *(f3 F3)*, respectively. These agents will go through the gap at *(d C)*.

Figure 4 shows the template generated after the first round of problem solving. Performance data (in this example, time taken and current drawn) recorded by the executors of the agents are included. In this example, agent α detected the conflict at *(d C)*, and resolved it using the strategy of replacing tasks, with the tasks to be replaces determined

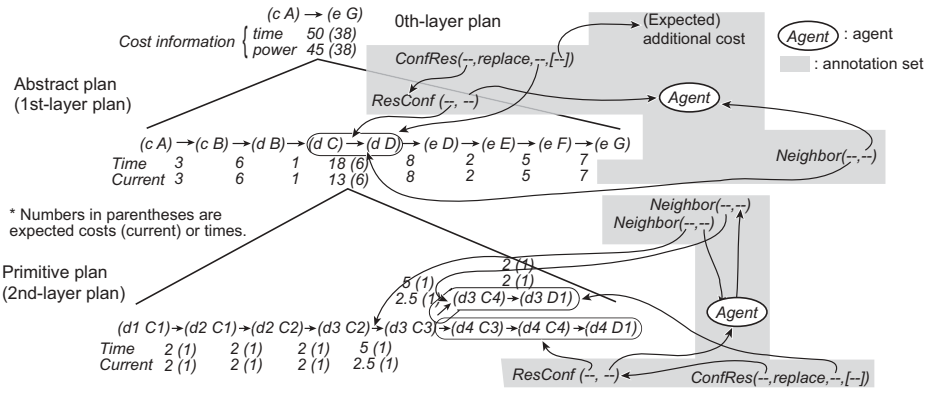


Fig. 4. Template after the first round of problem solving

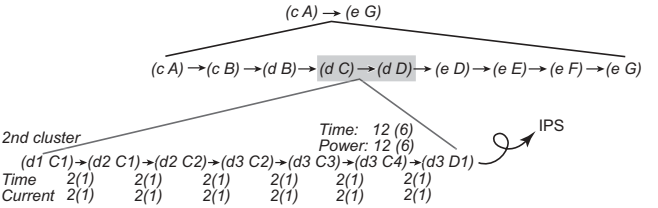


Fig. 5. Part of a template with cost data

by negotiation process. The routes selected by the two agents are shown in Fig. 3. After execution of its plan, agent α finds differences between the observed and expected performance data. So the learning procedure goes into the step of "data collection."

After solving a number of similar problems (that is, problems where the corresponding IPS is reused) the agent will find two clusters of data sets, for the gap-crossing IPS. One data set with have Time=18 and Current=13 while the other will have Time=12 and Current=12. The data sets in these clusters are shown in Figs. 4 and 5. None of the data sets are as expected. By comparing relative data, the agent can see that data sets in the first cluster are obtained only when two agents have neighbor relation while passing through the gap, while the second cluster is for situations where this does not apply.

A number of rules are then generated in the identification step; for example,

(r1) The expected performance for (d1 C1) $\rightarrow$ (d2 C1) are current = 2 and time = 2.

(r2) If another agent passes on the left- or right-hand side, the expected performance for task (d3 C2) $\rightarrow$ (d3 C3) are current = 2.5 and time = 5.

These rules concerning the hindrance relation are embedded in the corresponding IPS. They are applied when this part of the template is reused. As shown in Fig. 6, rule r2 as an annotation of the hindrance relation is added to the previous template.

It is also possible to add control to the agent so that it uses these rules to more actively explore other routes. For example, since r1 makes it aware of the extra cost,

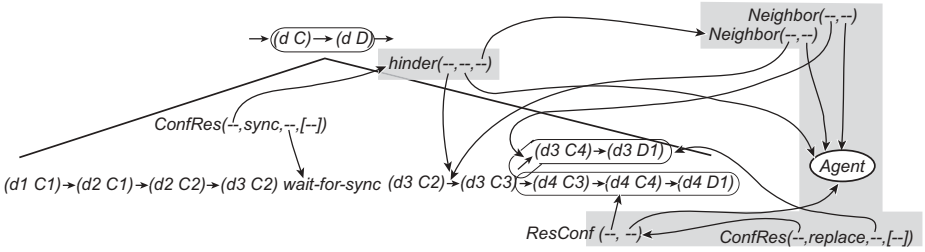


Fig. 6. Template after the application of rule r2

during planning to try to find a detour with a lower cost, such as shown in Fig. 3; or, r2 might make it select another resolution method such as synchronization (Fig. 6), which was not selected previously, because the planner noted that this method led to a wait of a few ticks while the other agent passed through the gap, but did not consider that the replacement method might lead to extra cost.

In the next example, we add the condition that the battery of each agent is shared between the motor driving its wheels and the CCD camera it uses to detect obstacles. Agent α's battery is dodgy so that it drops slightly when α goes up the slope around the lower gap in Fig. 3 (this slope was not included in the domain model, so the planner cannot anticipate this deficit relation) while also using its camera. In this situation, the plan in Fig. 4 fails at the task (d3 C2) $\rightarrow$ (d3 C3). Note that, for the sake of simplicity, we assume that only α is subject to this failure, and that the other agents do not fail, even when extra costs are incurred, as was the case in the previous example. This agent will experience the failure a few times, because it does not have knowledge of the fault. So, along with the results of the previous steps, agent α (and only agent α) will extract this rule concerning the deficit relation:

(r3) If there is another agent to the left or right, task (d3 C2) $\rightarrow$ (d3 C3) will fail.

The other agents will only generate the rule r2.

Agents other than α might then select the plan in Fig. 4 if it is not a time- and cost-critical situation (since, despite the extra costs, this plan will be successful costs). However, α will not agree with this resolution and will find another route or apply another resolution method, such as synchronization. In the case where the other agents are highly cooperative and α has a critical deadline, the other agents will give way to α and will thus explore other routes or apply *synchronization*.

7 Discussion and Conclusion

In this paper, we have looked at the storage and analysis of the hierarchical planning results from the past to identify implicit costs and resource relationships between activities in MAS contexts. In a plan-reuse framework, stored plans (templates) can be seen as accumulations of past experience. The proposed method takes advantage of such templates

by analyzing the data on locations, resources, costs, and timing that they contain and then extracting, as rules, descriptions of implicit relationships that may lead to resource conflicts. The plan-reuse framework also provides a guide to when detected differences in expected performance should be taken into account in planning. Finally, we gave an example that illustrates how such relationships are extracted and applied. Currently, we are implementing the Implicit Interaction Identifier as a subcomponent of the plan-reuse framework so that we can evalute our learning procedure.

A drawback of this framework is that, while the same task sequence may appear in solving different problems, they will be stored to different positions of the template. For example, the rule in relation to task (d3 C2) $\rightarrow$ (d3 C3) extracted in the previous section might apply in another situation where the same task has been selected as part of the plan, but the rule will be rediscovered rather than adopted. In this sense, the method proposed here is similar to self-monitoring to remedy inappropriate coordination [5] or learning appropriate coordination strategies in a situation-specific manner [7,10]. However, this work has a stronger focus on learning potential, implicit relationships. From another viewpoint, plan templates can be seen as (emerging) conventions by which multiple agents work together well (for example, [2] or Chapter 9 in [11]). The learned rules provide an additional mechanizm for selecting better social rules/conventions and thus improving multiple and/or joint activities.

References

1. J. F. Allen, "Maintaining knowledge about temporal intervals," *Communications of the ACM*, Vol. 26, No. 11, 832–843, 1983.
2. R. Alterman and A. Garland, "Convention in Joint Activity," *Cognitive Science*, Elsevier Science Inc., Vol. 25, No. 4, 611–657, 2001.
3. K. Erol J. Hendler, and D. S. Nau, "HTN planning: Complexity and expressivity," *Proc. of the AAAI94*, Vol. 2, 1123–1128, 1994.
4. M. P. Georgeff and A. L. Lansky, "Procedural knowledge," *Proc. of the IEEE Special Issue on Knowledge Representation*, Vol. 74, 1383–1398, 1986.
5. B. Horling, B. Benyo, and V. Lesser, "Using self-diagnosis to adapt organizational structures," *Proc. of the Int. Conf. on Autonomous Agents*, 529–536, ACM Press, 2001.
6. S. Kambhampati, "Supporting flexible plan reuse," In S. Minton, editor, *Machine Learning Methods for Planning*, 397–434. Morgan Kaufmann, 1993.
7. M. N. Prasad and V. R. Lesser, "The use of meta-level information in learning situation specific coordination," *Proc. of the IJCAI97*, 640–646, 1997.
8. E. Sacerdoti, "Planning in a hierarchy of abstraction spaces," *Artificial Intelligence*, Vol. 5, No. 2, 115–135, 1974.
9. T. Sugawara, "Reusing past plans in distributed planning," *Proc. of the Int. Conf. on Multi-Agent Systems (ICMAS95)*, 360–367, 1995.
10. T. Sugawara and V. Lesser, "Learning to improve coordinated actions in cooperative distributed problem-solving environments," *Machine Learning*, Vol. 33, No. 2/3, 129–153, 1998.
11. M. Wooldridge, *An Introduction to MultiAgent Systems*, John Wiley & Sons, 2002.

Evolutionary Learning of Multiagents Using Strategic Coalition in the IPD Game

Seung-Ryong Yang and Sung-Bae Cho

Department of Computer Science, Yonsei University
134 Shinchon-dong, Sudaemoon-ku, Seoul 120-749, Korea
{saddo,sbcho}@candy.yonsei.ac.kr

Abstract. Social and economic systems consist of complex interactions among its members. Their behaviors become adaptive according to changing environment. In many cases, an individual's behaviors can be modeled by a stimulus-response system in a dynamic environment. In this paper, we use the Iterated Prisoner's Dilemma (IPD) game, which is a simple model to deal with complex problems for dynamic systems. We propose strategic coalition consisting of many agents and simulate their emergence in a co-evolutionary learning environment. Also we introduce the concept of confidence for agents in a coalition and show how such confidences help to improve the generalization ability of the whole coalition. Experimental results show that co-evolutionary learning with coalitions and confidence can produce better performing strategies that generalize well in dynamic environments.

1 Introduction

In biology, co-evolution refers to the evolution of multiple species that affect one another. As one species evolves it changes the relationship that it has with surrounding species. In game theory and evolutionary game, the Iterated Prisoner's Dilemma (IPD) game and its variants have been used widely in modeling various social and economic phenomena [1][2]. It has also been used to study co-evolutionary learning under various conditions, where the primary purpose is not to model a dynamic system but to study how co-evolution can be used in learning novel strategies. For example, co-evolutionary learning of strategies for N ($N > 2$) player IPD games has been studied extensively [3][4][5]. Recently, a new approach to automatic design of modular learning systems has been proposed based on a combination of co-evolutionary learning and fitness sharing [6][7]. The IPD games with multiple choices of cooperation levels and reputation for each player have introduced a new dimension to the study of co-evolutionary learning and brought the IPD game one step closer to the real life situations [8].

This paper investigates the issue of generalization in a co-evolutionary learning system using the IPD game as an example. This issue was first closely examined in [9] using the classical 2-player IPD game. In this paper, we will show how the generalization ability of a coalition can be improved through player's confidence.

J. Lee and M. Barley (Eds.): PRIMA 2003, LNAI 2891, pp. 50–61, 2003.

Table 1. Axelrod's payoff matrix for the 2-player IPD game. It must satisfy following conditions: $T > R > P > S, \quad 2R > T + S$

		Opponent's move	
		Cooperation	Defection
Player's move	Cooperation	Player: R (3 point)	Player: S (0 point)
		Opponent: R (3 point)	Opponent: T (5 point)
	Defection	Player: T (5 point)	Player: P (1 point)
		Opponent: S (0 point)	Opponent: P (1 point)

The rest of this paper is organized as follows: Section 2 introduces the IPD game and the co-evolutionary approach to it. The representation of the IPD game strategy is given. Section 3 explains strategic coalition in the IPD game and how they are formed in evolution. Section 4 presents the experimental results. Finally, Section 5 concludes the paper with a brief summary and a few remarks.

2 Evolutionary Approach to the IPD Game

2.1 Iterated Prisoner's Dilemma Game

In the 2-player IPD game, each player can choose one of the two choices, defection (D) or cooperation (C). Table 1 shows the payoff matrix of the 2-player IPD game. The game is non-zerosum and non-cooperative. The game can be repeated infinitely. None of the players know when the game is supposed to end. According to the conditions given in Table 1, if the game is played for one round only, the optimal strategy is definitely defection. However, if the game is played for many rounds, mutual defection may not be the optimal strategy. Mutual cooperation will perform better than mutual defection for the IPD game.

2.2 Representation of Strategies

The representation of IPD strategies affects various aspects of evolutionary study, including the distribution of behaviors in the initial randomly-chosen population of players and the manner in which recombination and mutation create new players. Several different types of representation have been proposed in the IPD game such as finite state machine, logic tree, If-Skip-Action (ISAc) [10], Markov chain, and neural nets. However, the bit representation is generally used for the convenience of computer-based simulation. Figure 1 describes an example of bit string representation scheme used in this paper. The evolutionary approach to strategy learning for the 2IPD game was popularized by Axelrod [11].

2.3 Evolution of Strategies

In the 2IPD game, players are randomly selected from the population and then play the game. Generally, IPD game is composed of several processes such as

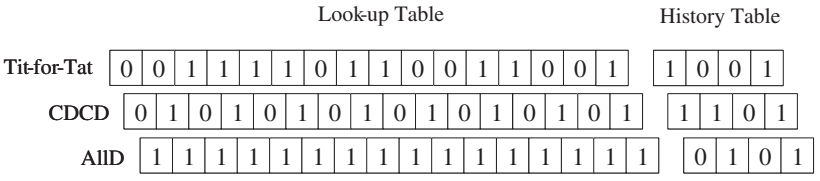


Fig. 1. Bit string representation examples

selecting action, moving and updating history. After enough games are played in a generation, each player's fitness is evaluated according to his total score obtained during the games.

3 Evolving Coalition of Strategies

Evolved strategies may overfit to their current population and perform poorly against good strategies that are not present in the current population [7][9]. One effective approach to address this problem and improve generalization ability of co-evolutionary learning is to automatically learn a group of cooperative strategies, each of which is a specialist in dealing with a certain type of opponents, so that the whole group becomes very robust and generalizes well [6][7]. Since such groups and specialists within a group are not partitioned manually but evolved, this approach can be regarded as a method of automatic design of modular systems, where each module consists of similar specialists. In this section, we define the strategic coalition and then how to form coalition.

3.1 Strategic Coalition

Cooperation among autonomous agents may be mutually beneficial even if the agents are selfish and try to maximize their own expected payoffs [12]. Coalition formation among those agents is an important method for cooperation in multi-agent environment [13]. For example, David and Jeffrey *et al*. have applied coalition formation to model future scenarios in international military coalition as a DARPA project [14][15].

To form coalition among agents, two main principles must be resolved in advance. One is which procedure a group of autonomous agents should use to coordinate their actions and cooperate, that is, how they form a coalition? The other is that among all possible coalitions, what coalition will form, and what reasons and processes will lead the agents to form that particular coalition [12].

In the real society, strategic coalition like this game can be shown easily between individual and individual, individual and group, and group and group. Let $I = \{A_1, A_2, \ldots, A_n\}$ be the collection of individuals in a population that participate in the IPD game. Let $C = \{C_i, C_j, \ldots, C_k\}$, $|C| \geq 2$ be the strategic coalition that can be formed among players. The coalition C is the element of

the individual group I: $C \subseteq I$, $|C| \leq |I|$. Every player has his own payoff, p_i, that earns from the IPD game against his opponents. The coalition has the vector, $C = \langle S_C, N_C, f_p, D_C \rangle$. Here, S_C, N_C, f_p, D_C mean the sum of a coalition payoff, the number of agents in the coalition, payoff function, and decision of the coalition, respectively. Now we can define the strategic coalition as follows.

Definition 1. *Coalition Value: Let w be the weight vector of each player's payoff. The payoff of coalition C_p is the average payoff by corresponding confidence of players that participate in the coalition.*

$$S_C = \sum_{i=1}^{|C|} p_i \cdot w_i \tag{1}$$

$$\text{where} \quad w_i = \frac{p_i}{\sum_{i=1}^{|C|} p_i}$$

$$C_p = \frac{S_C}{|C|}$$

Definition 2. *Payoff Function: Agents in the coalition get payoff with a given function. In general, 2-player IPD game follows Axelrod's payoff table.*

Definition 3. *Coalition Identification: Each coalition has its own identification number. This number is generated when the coalition is made with a given condition. This number may be removed when coalition exists no more. This procedure is made autonomously according to evolutionary game process.*

Definition 4. *Decision Making of Coalition: Coalition must have one decision that is made by participants in the coalition. The decision reflects the intention of the whole participants. We have used the weighted voting method for decision making of the coalition in this paper. Decision making of the coalition D_C is determined by function with payoff and confidence.*

$$D_C = \begin{cases} 0 \quad \text{(Cooperation)} & \text{if} \quad 1 < \frac{\sum_{i=1}^{|C|} {C_i}^C \cdot w_i}{\sum_{i=1}^{|C|} {C_i}^D \cdot w_i} \\ 1 \quad \text{(Defection)} & \text{if} \quad 0 < \frac{\sum_{i=1}^{|C|} {C_i}^C \cdot w_i}{\sum_{i=1}^{|C|} {C_i}^D \cdot w_i} \leq 1 \end{cases} \tag{2}$$

Definition 5. *Payoff Distribution: The agents in the coalition get payoffs from the game against another player or coalition. These payoffs must be distributed to each agent with a given function according to its confidence.*

$$p_i = w_i \frac{S_C}{|C|} \tag{3}$$

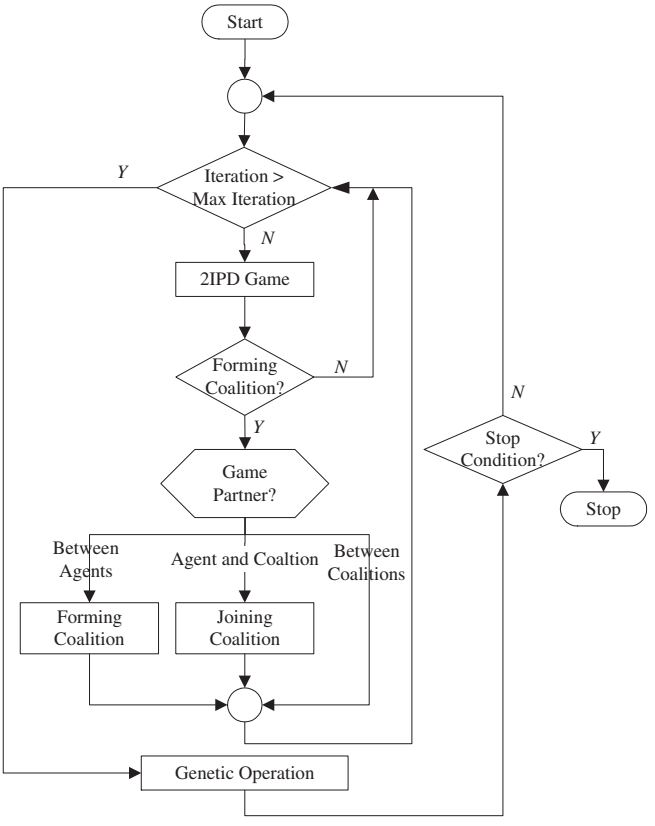


Fig. 2. The 2-player IPD game with coalitions

3.2 Coalition Formation

Coalition formation includes three activities: (1) coalition structure generation, (2) solving the optimization of each coalition, and (3) dividing the value of the generated solution among agents [16]. While the IPD game between players proceeds, each player checks the opponent's intention to form coalition. At this time, the payoff is an important factor to form coalition. In this paper, we propose a different method for evolving coalition. It uses the idea of confidence. A player's confidence is not pre-defined and fixed, but evolved. A player may have different confidences in dealing with different opponents. A player is allowed to take part in deciding the next move of coalition at the rate of his confidence. In other words, coalition's next move is determined by players' confidences.

The IPD game with coalitions is played according to the procedure given in Figure 2. During each generation of evolution, the 2-player game is played between two randomly selected players. After each game, each of the two players considers making (or joining) a coalition to get more payoffs. If all conditions

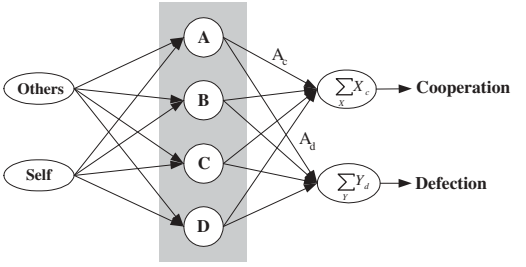


Fig. 3. Combining the confidences of multiple players within a coalition

are satisfied, they make a coalition. As the game is played over and over again, there may be many coalitions that players make. Therefore a player can play a game against a coalition. A coalition can also play against another coalition. Combining coalitions is not considered in our study. There is no action after a game between coalitions.

Once a player has joined a coalition, players within the coalition (including the new one) will play IPD games in round-robin. For k players in a coalition, a total of $k(k\text{-}1)/2$ games will be played. If the total number of players (i.e., k) is greater than a pre-defined maximum coalition size, the weakest player (in terms of the total payoff he/she obtained in all round-robin games) will be removed from the coalition. All players within a coalition are ranked (sorted) according to the total payoff. A player's confidence is assigned in proportion to his/her ranking in the coalition. Figure 3 shows how to combine the confidences of different players in a coalition to produce the coalition's next move. In the figure, **Others** indicates opponent's move and **Self** indicates coalition's move in previous steps. **A**, **B**, **C** and **D** are players within a coalition and $\mathbf{A_d}$ is player's next move, reflected by their confidences, for a given history. For example, if **A**'s next move is cooperation, $\mathbf{A_c}$ is $1\times$**A**'s confidence and $\mathbf{A_d}$ is 0. There is no weight for inputs to a player.

3.3 Evolution of Coalition

Coalitions may be formed or removed from population during evolution. In our experiments, coalition below the average fitness of players in population is removed. In addition to coalition formation as described previously, new coalition may also be generated by crossover of existing coalitions. Crossover of coalitions is done by exchanging the same number of players, as shown in Figure 4. It is important to point out that a player will only exist in the coalition once it has joined in. As a result, weak players may be in the population without belonging to any coalitions (separate from the population) while strong players join coalition. To remedy this problem, new players in the population are generated by recombining players from coalitions. We do not use weak players in the population to generate new players.

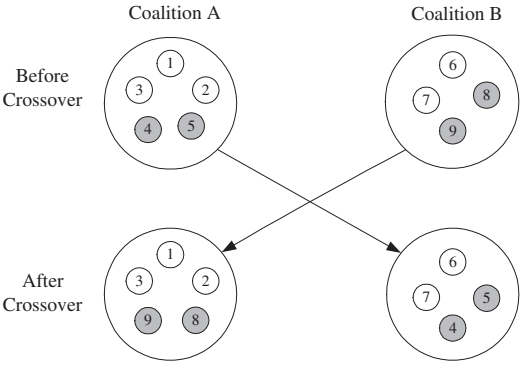


Fig. 4. Crossover used in evolving coalitions

3.4 Gating Strategies in Coalitions

To improve the generalization ability of a coalition, techniques such as opponent modeling and gating can be used. Opponent modeling predicts opponent's next move. Such prediction, if sufficiently accurate, can help the coalition to choose the optimal move. However, it is difficult to model opponent's strategy accurately. In this paper, we use player's confidence to decide which move should be taken by the coalition against an opponent.

A population consists of coalitions that have a different confidence table. A coalition has the same players that other coalitions have. Here, good players are obtained during the evolution. A coalition has a history table for its own and opponent's moves and can use the information to change player's confidence dynamically. The confidence is first randomly initialized as real numbers between 0 and 2 because the average confidence is 1. All coalitions have the fixed number of players that are extracted from coalitions in the last generation of the co-evolutionary process but a different confidence table. If a coalition assigns confidences effectively to the players in a given history, it may get a higher score than the others. In this paper, confidence is represented as real numbers and can be changed by mutation. A new confidence table may also be generated by crossover of confidence tables. Crossover is done by changing confidences at crossover point between coalitions keeping the information of players maintained. The training set for adjusting player's confidence consists of several well known game-playing strategies, such as TFT, Trigger, CDCD, etc [17][18]. During evolution, appropriate confidences are determined from coalitions. Crossover mixes the confidences between coalitions in the population, and mutation changes a specified confidence into a random real number between zero and two.

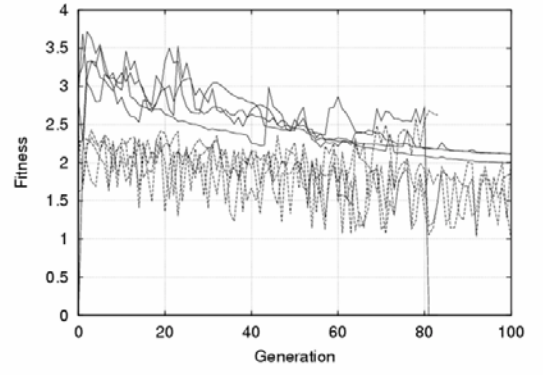


Fig. 5. Average fitness of coalitions and players in the population. Solid lines are for coalitions and dashed lines for players

4 Experimental Results

The first experiment is designed to evolve coalitions using co-evolutionary learning and the second is to evolve player's confidence so that coalitions can play adaptively against different opponents. In our experiments, the payoff matrix used follows the one given in [9].

4.1 Evolving Coalition

In our experiments, we use the population size of 50, crossover rate of 0.6, mutation rate of 0.001 and rank-based selection. A genetic algorithm with one-point crossover and elitism is adopted [19]. For the 2-player game, the history length is 2. The maximum coalition size is 10 and the maximum number of coalitions is one third of the population size (i.e.,16). For the experiments presented here, player's confidence is fixed and not evolved. The evolution of player's confidence will be considered in the next subsection.

Figure 5 shows the average fitness of coalitions for 4 runs during evolution. In the beginning of evolution, the average fitness of coalitions is above that of individual players in the population. However, the difference decreases as time goes by, because players in the population have gradually learned to deal with stronger coalitions. (Notice that new players in the population were generated by recombining strong players from coalitions.) In other words, players in the population also gradually evolve to adapt to their environment.

In the above experiments, the coalition's next move is determined by players' weighted voting at a fixed rate (weight here corresponds to confidence). This means that a coalition's strategy against an opponent can be represented by a single player. A coalition may not be as fit as it should have been in comparison with individual players not in the coalition. Figure 6 shows the number of coalitions during evolution. The number of coalitions could become very small or even zero as a result of fixed player confidence.

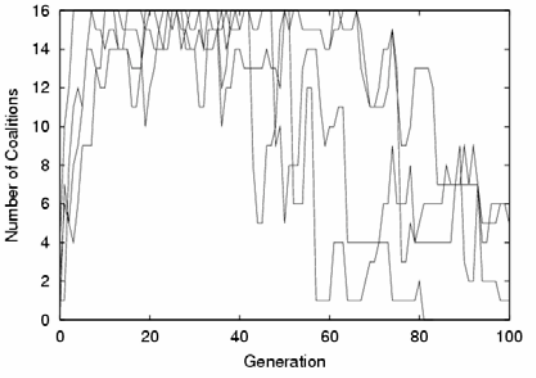


Fig. 6. The number of coalitions in the evolution

Table 2. Training set for evolving player's confidence

Strategy	Characteristics
TFT	Initially cooperate, and then imitate opponent's action
TF2T	Similar TFT, but defect for opponent's 2 defection
Trigger	Initially cooperate. Once opponent defects, continuously defect
AllD	Always defect
CDCD	Cooperate and defect over and over
CCD	Cooperate and cooperate and defect
C10Dall	Cooperate before 10 rounds and always defect after that
Random	Random move

4.2 Evolving Confidences

In this experiment, we use the same experimental setup as before. The training set used in evolving player's confidence consists of seven well-known strategies and a random strategy, as shown in Table 2. Figure 7 shows the average fitness of evolved coalitions.

To evaluate the generalization ability of coalitions with evolving confidences, we have selected 30 top ranked players in a random population of 300 as the test set. Each player in the test set then plays against evolved coalitions in round-robin games. We have experimented it for 10 runs to show the result of evolving confidences. In the test of generalization ability of coalitions, player's confidence is allowed to vary with different opponents. Table 3 shows the performance of a player against the 30 test players. The second and third columns in this table indicate that there is a significant improvement in coalitions' performance when player's confidence is allowed to evolve. The second column indicates the results without the evolution of confidences and the third column indicates those with it. In general, coalitions' winning percentage has increased significantly and the losing percentage decreased. In comparison with players in the training set, the coalitions also perform quite well against the 30 test players. They have only

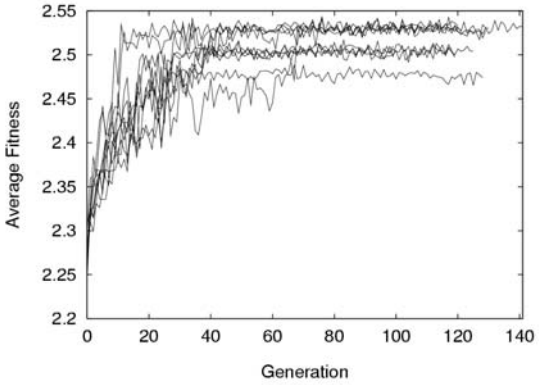


Fig. 7. Average fitness of coalitions

Table 3. Performance against opponent strategies

Strategy	Wins	Ties	Avg.	Opp. Avg.
Before	8.64±4.9	6±2.19	1.84±0.28	1.75±0.59
After	18.55±0.5	4±0.63	2.16±0.07	0.92±0.29
TFT	8	0	1.70	1.77
Trigger	30	0	2.13	0.80
TF2F	7	0	1.54	2.40
AllD	30	0	2.17	0.7
CDCD	0	0	1.05	2.75
CCD	0	0	0.91	3.34
C10Dall	27	0	1.97	1.12

performed worse than AllD on average. They seem to be more likely to tie a game than those players in the training set. This turns out that the coalitions have learned not to lose a game as its top priority. Figure 8 shows the payoff comparison of the evolved strategy with each test strategy.

5 Conclusions

Combining multiple players in a group can be a very effective way of designing a learning system that generalizes well. We have evolved coalitions consisting of a number of players in the IPD game. Each player in a coalition has a confidence which specifies how well he/she is in dealing with an opponent. The confidence is not fixed. It is different for different opponents.

Experiments have been carried out to compare players with and without adaptive confidences. It is found that using adaptive confidences can improve coalitions' performance in testing significantly. Coalitions with adaptive player's confidence can deal with very different opponents, regardless of whether they are cooperators.

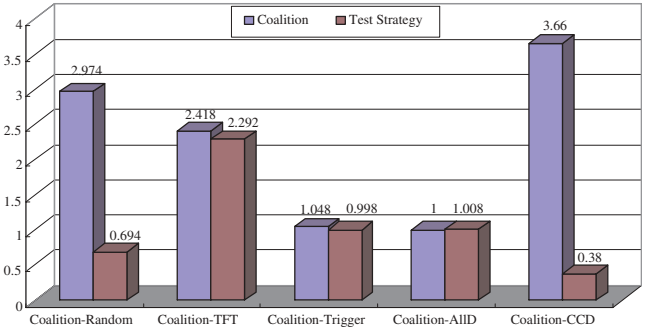


Fig. 8. Payoff comparison of the evolved strategies with each test strategy

Although we have used the 2-player IPD game in this paper, some of the results we have obtained may be applicable to more complex games. It will be very interesting to study the evolution of coalitions and confidences in a real world environment. For example, it is interesting to investigate how coalitions can be formed among different countries in the world, how coalitions can be formed among different parties in a country, how coalitions can be formed in a commercial market, etc.

Acknowledgements. This work was supported by Korea Research Foundation Grant(KRF-2002-005-H20002).

References

1. Ord, T., Blair, A.: Exploitation and peacekeeping: Introducing more sophisticated interactions to the iterated prisoner's dilemma, *Proceedings of the 2002 Congress on Evolutionary Computation*, **2** (2002) 1606–1611
2. Tesfatsion, L.: Agent-based computational economics: Growing economics from the bottom up, *Artificial Life*, **8** (2002) 55–82
3. Yao, X., Darwen, P. J.: An experimental study of N-person iterated prisoner's dilemma games, *Informatica*, **18** (1994) 435–450
4. Seo, Y. G., Cho, S. B., and Yao, X.: Exploiting coalition in co-evolutionary learning, *Proceedings of the Congress on Evolutionary Computation 2000*, **2** (2000) 1268–1275
5. Fletcher, J. A., Zwick, M.: N-Player prisoner's dilemma in multiple groups: A model of multilevel selection, *Proceedings of the Artificial Life VII Workshops*, Portland, Oregon (2000)
6. Seo, Y. G., Cho, S. B., and Yao, X.: The impact of payoff function and local interaction on the N-player iterated prisoner's dilemma, *Knowledge and Information Systems: An International Journal*, **2(4)** (2000) 461–478
7. Darwen, P. J., Yao, X.: Speciation as automatic categorical modularization, *IEEE Transactions on Evolutionary Computation*, **1(2)** (1997) 101–108

8. Yao, X., Darwen, P. J.: How important is your reputation in a multi-agent environment, *Proceedings of IEEE International Conference on Systems, Man, and Cybernetics (SMC'99)*, **2**, IEEE Press, Piscataway, NJ, USA (1999) II-575–II-580
9. Darwen, P. J., Yao, X.: On evolving robust strategies for iterated prisoner's dilemma, *Progress in Evolutionary Computation, Lecture Notes in Artificial Intelligence*, **956**, Springer-Verlag, Heidelberg, Germany (1995) 276–292
10. Ashlock, D., Joenks, M.: ISAc lists, a different representation for program induction, *Genetic Programming 98, Proceedings of the Third Annual Genetic Programming Conference*, Morgan Kaufmann, San Francisco (1998) 3–10
11. Axelrod, R.: The evolution of strategies in the iterated prisoner's dilemma, *Genetic Algorithms and Simulated Annealing*, **3**, Morgan-Kaufmann, San Mateo, CA (1987) 32–41
12. Shehory, O., Kraus, S.: Coalition formation among autonomous agents: Strategies and complexity, *Fifth European Workshop on Modeling Autonomous Agents in a Multi-Agent World*, Springer-Verlag, Heidelberg, Germany (1993) 56–72
13. Shehory, O., Sycara, K., and Jha, S.: Multi-agent coordination through coalition formation, *Agent Theories, Architectures, and Languages* (1997) 143–154
14. Allsopp, D. N., Kirton, P., Bradshaw, M., et al.: Coalition agents experiment: Multiagent cooperation in international coalitions, *IEEE Intelligent Systems*, **17** (2002) 26–35
15. Tate, A., Bradshaw, M., and Pechoucek, M.: Knowledge systems for coalition operations, *IEEE Intelligent Systems*, **17** (2002) 14–16
16. Sandholm, T. W., Lesser, V. R.: Coalitions among computationally bounded agents, *Artificial Intelligence*, **94** (1997) 99–137
17. Axelrod, R.: *The Evolution of Cooperation*, New York: Basic Books (1984)
18. Axelrod, R., Dion, D.: The further evolution of cooperation, *Science*, **242** (1988) 1385–1390
19. Goldberg, D. E.: *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Reading, MA (1989)

Multi-agent Travel Planning through Coalition and Negotiation in an Auction

Ming-Chih Hsu, Paul Hsueh-Min Chang, Yi-Ming Wang, and Von-Won Soo

Department of Computer Science
National Tsing Hua University
101, Kuang Fu Rd, Sec.2
HsingChu, Taiwan 300 R.O.C
{scat,pchang,vicnest,soo}@cs.nthu.edu.tw

Abstract. In a travel scenario the travel agent often faces a situation: the users only provide their preferences on visiting destinations, while the visiting orders and the arrangements of transportations are left to the decision of the travel agent. Therefore the travel agent must find suitable (both efficient and economic) tracks for the users given their visiting destinations, different transportation services, constraints of users such as time, budget, and preferences. Although the service route map of each transportation company can be derived beforehand, the negotiable price information is often private. It is not feasible for a user or a travel planning agent to determine the total expense by simply summing up the list prices of all transportation track segments, and hence the selection of the most efficient track is also not possible. One way to find out the best route is to provide a mechanism for the transportation companies to form coalition and negotiate on the prices based on their own utilities and profit concerns. In this paper we propose a mechanism to solve the best tourist track problem. The mechanism includes a heuristic shortest path finding algorithm for a track graph and a track winner determination auction, called Z-auction, for track competition. We show how the travel planning problem can be solved through the multi-agent coalition and negotiation in the Z-auction under the multi-agent problem solving environment.

1 Introduction

1.1 The Goals and Tasks of an Online Travel Agent

In real world, "a travel agent's goal is the best possible trip for each client", said American Society of Travel Agents President and CEO Richard M. Copland [1]. For an on-line travel agent, his goal should be the same with the one in the real world Once a user tells an on-line travel agent his travel plan and personal preferences, the agent will decompose his best-possible-trip goal into many subtasks such as scenic spots selecting, track planning, accommodation bookings, flight reservations, and car rentals, etc.

For clarity, in the following texts, the term "travel agent" means the on-line travel agent unless the word "human" precedes it. The tourist track is abbreviated as the

J. Lee and M. Barley (Eds.): PRIMA 2003, LNAI 2891, pp. 62–73, 2003.

track. The track segment refers to the route between any two cities within a tourist track and is abbreviated as a segment.

1.2 The Best Tourist Track Problem

When a travel agent serves naïve travelers (the user), she often faces a problem that the user might only know the general destinations that are attractive but do not know specially the details. Such users have no ideas about the sequence of tourist tracks and the arrangements of suitable transportations. When serving naïve users, a travel agent must find the best tourist track and the most budget transportation in every segment of the track. For example, Steve in Taipei, Taiwan heard that Tokyo and Osaka in Japan are both interesting cities and he had a seven-day holiday, so he asked a travel agent to plan a tourist track to Tokyo and Osaka for him. The travel agent should arrange a travel package (including the visiting spots, hotels, and transportations) to Tokyo and Osaka, plan the possible sequence of tourist tracks, and decide a most suitable and economic track for Steve.

At first glance, the problem can be experience-dependent (travel agent knows which sequence is popular) or contract-dependent (if the travel agent is in contract with some transportation companies, he will utilize the route combined by these companies). But in fact, the possible solution combinations can be many because:

1. Several travel packages with different durations may exist in one city.
2. More than one transportation tool and company may run between two cities.
3. The sequence to visit the destination cities is undefined.
4. The time and cost do not guarantee the global optimal.

We worked out a heuristic scheme to solve the problem.

1.3 The Potential Problems of Online Travel Agent

A professional human travel agent knows how to get travel-related information, how to communicate with customers, knows which places are popular, and is good at bargaining with service providers. Even if he needs information from internet, he filters out irrelevant information. For an on-line travel agent, however, we expect him to find more information via internet and filter out the insignificant information since human tends to miss related information due to carelessness. But due to the enormous quantity of information and the dynamic changes of information, even the most powerful on-line travel agent cannot collect all the data. Without the help from other kinds of agents, an on-line travel agent may fail to find the most suitable services because of the dynamic changes of service providers.

1.4 A Tourist Multi-agent Model

In current web services, most services are passive. It means that the services are waiting for service requests. However, with intelligent agents, services can be provided actively.

Roles in our tourist multi-agent model include:

1. Travel agents: responsible for user agents' requests and post the requests in valid forms so that other service agents know how to fulfill them.

2. City tourist guide agents: each city in real world have tour guides in charge of the tourist business such as arranging travel packages that are favored by the travelers. A travel package may consist of local attractive spots, accommodations, transportations during the tracks, and the itinerary of the whole tourist plan. City tourist guide agents could arrange different types of travel packages according to seasons, or change the lengths of travel packages to satisfy diversified demands from different travelers.

3. Transportation service agents (abbreviated as the transportation agents): transportation companies who carry customers between cities are the main agent roles of this paper. Transportation agents could get the information of the travel packages from travel agents requested possibly by certain travelers. Because the sequence of travel packages is not specified, transportation agents could tell the travel agent the transportation type and the list price they could offer if he can provide the transportation between any two travel packages. A transportation agent may give discounts to increase the sales so that he could fully utilize the capacity of the transportation he owns.

The tourist multi-agent model establishes a competition free market that helps travel agents to find the most suitable and economic track for their traveling customers.

2 The Scenarios and the Design of the Tourist Multi-agent System

2.1 The Scenarios of Tour Planning

Imagine a traveler Tom from Tokyo, Japan wants to spend 20 days in South Korea. He chooses 5 cities and the travel packages are gathered by a travel agent from city guide agents as the destinations. And then he specifies the starting location, the end location, the starting time, the end time, the budget, and the time value. The time significance value indicates how important and urgent the traveling time is to the traveler. The transportation agents submit the available routes and their list prices once the travel agent posts the cities and the staying period within each city. The travel agent plans candidate tour tracks (filter out impossible tracks that exceed time constraints or the user budget) and holds an auction for these tracks so that the transportation agents can compete for their favorite tour tracks (In general, a transportation agent should bid for the tour track in which he would earn the most profit). At the auction closing time, the winner tour track is announced. The winner track should be the one that best suits Tom's budget and won't keep Tom waiting too long for the transportation if the time significance value he specified is high. But if Tom is at leisure and the time significance value he specified is low, the trip plan can be a relaxed one.

2.2 Design of the Tourist Multi-agents System

To reduce the complexity of complete tourist multi-agent system described above and focus on the interactions between transportation agents and travel agent, we make the following assumptions to reduce possible complexities of the problem:

1. All travel packages were pre-made and stored inside the city tourist guide agent.
2. The user would select one from the travel packages within one city and thus ensure staying the whole period in each city.
3. Once the transportation agent who wishes to bid a route service but doesn't have a direct route between any two cities posted by the travel agent, he could find other transportation agents to help him to connect the two cities. If a transportation agent could earn more profits by getting the route service via cooperation, he might cooperate with other transportation agents. We simplified the process of cooperating and coalition formation to provide the direct transportation between two cities by assuming if there is no transportation between two cities, then the two cities is disconnected.
4. The type of transportation is not concerned at the current stage. But chances are that only one type of transportation is available between two cities. For example, if a user insists on taking no airline between international routes, the travel agent may not be able find a feasible transportation for her.
5. The quality of the transportation is neglected. We use only a simple function to express user's preference over the price and the speed of the service provided by the transportation. Generally speaking, the transportation with higher speed is more expensive than the ones with lower speed. Below is a user preference function:

$$Z = X \times T + Y \times P \tag{1}$$

T is the sum of the transportation time plus the total waiting time for the transportation to the next destinations. And P is the total price of all transportations. X and Y are the weights of time and price respectively. In our interface, the user's preferences over the speed and the price are evaluated by the time value (T') bar from 1 to 100. X is the time significance value divided by 100 (i.e. T'/100), and Y is 1 minus X (i.e. 1 – T'/100). When the time significance value is low, the user doesn't care much about the speed of transportation, so Y is greater than X. While the time significance value is high, the user care much about the time spent, so X is greater than Y. For business travelers, they are time-sensitive, so their time value is high. But they should also be money-sensitive; the related work is in [2].

3 System Overview

We choose web service technology as the basic platform of our tourist multi-agent system. Each agent is composed of web services in addition to the ability of making decision and all agents register at our private UDDI server (jUDDI [www.juddi.org]). Agents can communicate and negotiate by using pre-defined port types to send and receive messages. We simulated the data of transportation companies, travel packages, and the behaviors of transportation agents in the experiments.

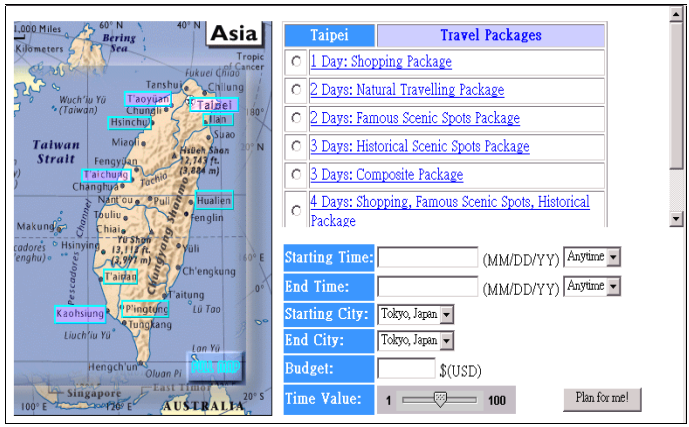


Fig. 1. User interface of our system

Figure 1 is a user interface in which user may choose travel packages when a city is selected. After user selected his destinations from the map [3] and specify his preferences, the result of the travel plan will be replied to the user by E-mail within reasonable time.

4 The Track Planning Algorithm for Travel Agents in the Markets

With the inputs Table 1 from user interface, the travel agent will conduct a track planning procedure to find out possible tracks with minimal stops. It consists of 8 stages.

Table 1. A track formation request with constraints and preferences from the user interface

Field name	Description
Travel date/time	Starting and end travel date/time
Budget	User budget
Start/End locations	User's Starting/End locations
Visiting destinations	User's destinations with staying times
Time-Price Preference	Preference z = x*time + y*price

Stage A

Announce user track requests to all attending transportation service agents, therefore each transportation agent can submit its qualified track segments (the transportation service connecting any two user selected locations) to the market. The submitted segments should contain the information in Table 2.

Table 2. Contents in a track segment

Field name	Description
Destinations	The starting location and end location
Start weekday(s)/time	Transportation weekday(s)/time
Duration	The time needed in the transportation
Price	The list ticket price

Upon receiving all the track segments from transportation agents, the market can generate a full track graph by joining all track segments as illustrated in Fig.2. Of course this is a directed graph. A possible travel track is a path that starts from the starting vertex (the user's starting location), visits all vertices (user's specified destinations except the start and ending locations) and terminates at the ending vertex (the user's ending location). The paths (travel tracks) with minimal number of edges are preferred because no one wants to waste time on transportation transfers.

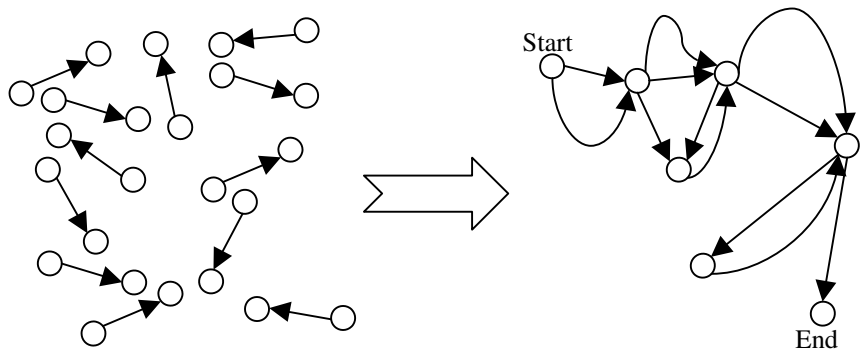


Fig. 2. Constructing a full track graph (right) from track segments (left)

Stage B

The simplest way to generate all possible paths given a directed graph and a starting vertex is to use a depth-first search with backtrack algorithm [4], but doing this will have a risk ending up in an enormously large search space (of course it's a NP problem).

A better way to deal with this problem should be use as much information as possible in the directed graph. The information includes vertices precedence, graph connectivity, and the topology of the track graph. By introducing the techniques that make use of the information, we can reduce the search space to a feasible degree.

First the edges with identical source vertices and destination vertices are combined. In other word, any two vertices will have one edge connecting them if there is at least one edge between them. And the lengths of all edges are set to 1. This process can further simplify the graph for later processing (Fig.3). Then a fully connected component algorithm [5] is used to determine all fully connected components in the directed graph. A fully connected component in a directed graph is defined as: for any

pair of vertices in this component these two vertices can reach each other in both directions. By finding fully connected components we can isolate vertices with weaker precedence requirements. Therefore if two vertices (A and B) belong to different fully connected components, then there must be only one precedence order between them (A before B or B before A, but not both). The resulting graph is illustrated in Fig.4. It is easy to show that by treating each fully connected component as a compound vertex, the newly formed graph is a directed acyclic graph (DAG). The partial order information (vertices precedence) can be derived from this DAG in stage C.

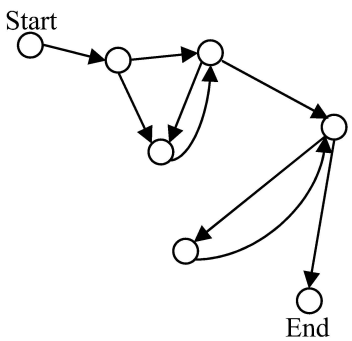


Fig. 3. Simplified track graph

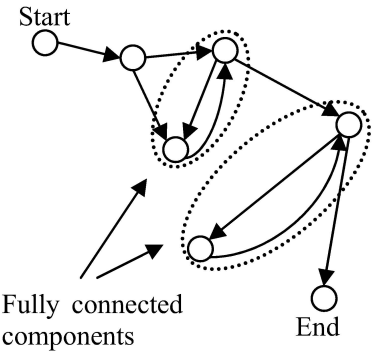


Fig. 4. Fully connected components marked graph

Stage C

Because our goal is to find the shortest path(s) that meet all vertices in this DAG, and we know that each path in a DAG does not meet any vertex twice; therefore if a topological order of a DAG is a path, it is the shortest path. And such a path is a Hamiltonian path (or Hamiltonian circle if the user wants to return to his starting location).

An exhaustive topological sorting algorithm is applied on the DAG to derive all topological orders of vertices in this DAG (Fig.5). By conducting a simple Hamiltonian path check algorithm on these topological orders, all possible shortest paths can be found.

We should remember that some vertices in the shortest paths derived above are compound vertices. Each compound vertex is in fact a fully connected directed graph. To derived true shortest paths, it is necessary to use a depth-first search with backtrack algorithm mentioned in step B (Fig.6). Although this may lead to large searching space, but after the above processes the search space of the shortest path of the DAG has been greatly reduced.

Because the edges in each path from the above process are in fact simplified representations of track segments (the segment information are neglected), an additional step is needed to enumerate possible tracks. Each edge in a path is replaced by corresponding track segments (Fig.7), and all possible tracks are generated by enumerating all possible track segment combinations in each track.

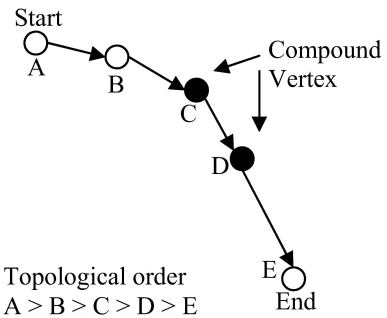


Fig. 5. Topological sorting

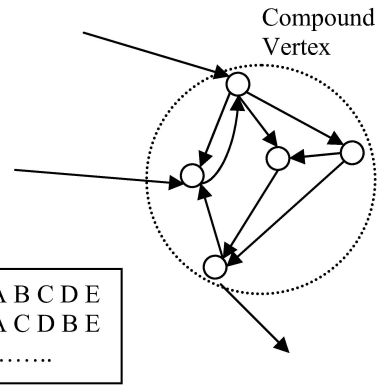


Fig. 6. Path generation for compound vertex

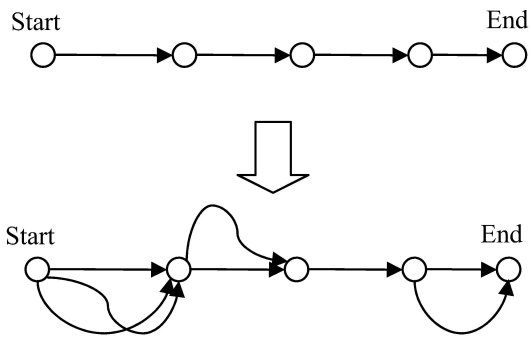
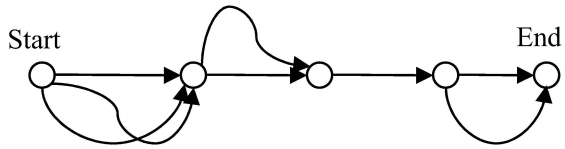


Fig.7. Track segment replacement

Stage D

It is possible that there are several alternatives for every segment of a track. And if we enumerate all possible segment combinations, the number of generated tracks will likely to be very huge. To protect against the possible storage overloading problems caused by a large number of tracks, a pruning algorithm is proposed (Fig.8). In this algorithm, the two segment combinations with the smallest z summations are selected for each track. This algorithm is set to activate when the total number of possible tracks exceeds a threshold. Although using this algorithm may falsely prune potentially good tracks, it is still worthwhile to protect the market from failing to function due to storage overload.

The final output of the track planning algorithm is a set of possible shortest tracks. Each track contains several segments provided by different transportation agents. The user related initial cost Z-value is calculated for each track. This Z-value will be used in the track competition auctions to further reduce the travel costs of users.



1. For each candidate track
Find the smallest z segments combination by selecting the smallest z segments in each hop in the track. The selected segment in each hop is marked. The smallest z path in each candidate track is derived.

2. For each candidate track
Find the second shortest segment combination by finding the hop whose remaining segment has the smallest-z segment difference with the marked smallest segment. The marked segment in that hop is replaced by the second smallest-z segment. The newly derived path becomes the second smallest-z in this candidate track.

3. The other possible segment combinations of each candidate track are abandoned.

Fig. 8. The pruning algorithm

5 Winner Selection Mechanism in the Track Competition Auction

After the track planning and filtering process above, a collection of candidate tourist tracks is chosen. Deciding the winner of the tracks is a two-level process: tracks are bid in an auction by the transport agents who negotiate the price on each track among themselves. This mechanism will allocate the track to those who deserve it the most in terms of minimum "cost" as the winner.

5.1 The Z-Auction

Tracks are bid at Z-auction which is a variant of English auction. Unlike traditional English auction that requires bidders to increase their bidding prices at each round of valid bid, the bidders in Z-auction must compete for the lowest Z-value. The Z-value is the user preference attached to each candidate track and it descends as the auction proceeds. The initial quote is the lowest initial Z-value of all candidate tracks. The auction time is divided into discrete time units (ΔT'*s*). The minimal decrement of Z-value is ΔZ, which means that the Z-value of a valid bid must be at least ΔZ less than the current quote. Those tracks that fail to low their Z-value quit the auction. The quote is updated after each ΔT pass. The lowest Z-value offered during the last ΔT period is the new quote. If there is no valid bid after ΔT passes, the auction closes, and the track who made the last valid bid wins.

5.2 Negotiation among Segment Owners in a Coalition

The auction mechanism above does not specify how the coalition of the transport agents decides the *Z*-value of a track. Since the start time and duration of each

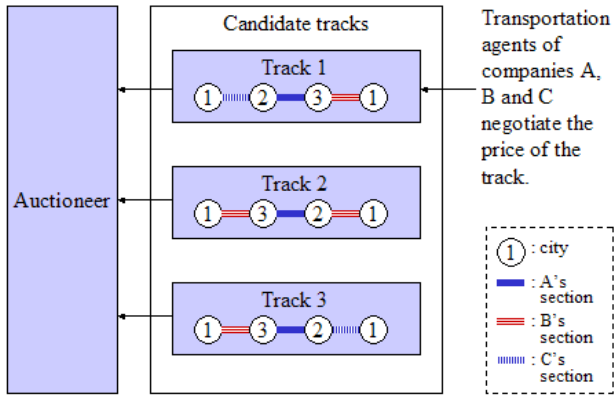


Fig. 9. The track selection mechanism

segment of a track have been fixed, the Z-value can only be affected by the prices of the segments. If one company owns all segments of a track, the company's transportation agent alone can decide whether or not to follow the bid. If the track contains segments from a coalition of multiple companies, their agents must negotiate how to lower the prices during the auction. Suppose track R contains track segments owned by companies A, B, and C, the current quote is Q, and the current Z-value of R is Z_R, $Z_R > Q$. Then R must lower its Z-value by $Z_R - Q + \Delta Z$ in order to bid in the auction. That means A, B and C must negotiate the sharing of the $Z_R - Q + \Delta Z$ decrement in the Z-value.

An auction-like negotiation mechanism is proposed here. Each ΔT is divided into n time slices. Let $\Delta t = \Delta T/n$. The minimum decrement $\Delta z = (Z_R - Q + \Delta Z)/n$ is calculated at the beginning of each ΔT. Any change of Z_R is revealed to A, B, and C, each of which also knows the Z-values of its own segments. During each Δt, at least one of A, B and C must lower its price by $\Delta z/Y$ so that the Z-value of the track drops by Δz, or the negotiation fails and the track quits the auction. If the negotiation does not fail until the end of a ΔT, the track will be able to make a valid bid in the auction. After that, the quote changes and Δz is recalculated, and the transport agents negotiate again, until either the negotiation fails at some point in time or the auction closes. This mechanism ensures that if there is still room for potential agreements among A, B and C, an agreement will always be made at least before the end of the ΔT, as long as Δz is small enough.

5.3 Negotiation Strategies of Transportation Agents

Transportation agents in the negotiation process can adopt many possible negotiation strategies. When multiple companies are present in a track, the company who needs the deal more will be willing to lower its prices more than others. Suppose track R contains segments owned by companies A and B, who are not present in other candidate tracks. Let the cost of segments owned by A and B be C_A and C_B respectively. Let the price of the segments owned by A and B be P_A and P_B. The rational behaviors of A and B are shown in table 3.

Table 3. The rational decision behaviors of A and B on whether to lower the price

At the end of each Δt	$P_A - \Delta z/Y > C_A$	$P_A - \Delta z/Y \leq C_A$
$P_B - \Delta z/Y > C_B$	Either *A* or *B* will lower the price. (Nondeterministic)	*A* will not lower the price. *B* will lower the price.
$P_B - \Delta z/Y \leq C_B$	*B* will lower the price. *A* will not lower the price.	Neither *A* nor *B* will lower the price. Negotiation fails.

When a company is present in multiple tracks, the company will wish the track that is the most profitable to win, and therefore will be reluctant to lower the price in the less profitable tracks. In other words, a company can make greater concession if it owns more segments in a track.

It is worth noting that in some cases there may be an "internal" agreement between two companies. The agreement makes it possible that a company's concessions in one situation in the auction will be compensated in other situations. For example, the track from city *C1* to city *C2* and then to city *C3* is a very popular one, and company *A* who dominate the segment service from *C1* to *C2* may demand company *B* to lower the price of the segment from *C2* to *C3* to form a competitive track together. A can compensate B's loss in other situations where *B* could demand *A* to lower the price. The strategy analysis in game-theoretical negotiation with compensation actions has been studied by our previous work [9].

6 Discussion

Our negotiation mechanism for transportation agents do not require each of them to reveal their true prices and preserves to some degree the privacy of the individual cost. For example, if a track contains segments from *A*, *B*, *C*, and *D*, the only prices *A* knows are its own price and the total price of the track, not the individual prices of *B*, *C* and *D*. However, if there are only two companies *A* and B in the track, *A* can easily guess *B*'s price by subtracting *A*'s price from the total price. This can be prevented by not revealing other companies' identities in the track during negotiation, but without that information their transportation agents cannot negotiate privately (for example, about an internal agreement.)

Sandholm [7] proposed an extended version of the Contract Net Protocol to solve the vehicle routing problem. In his scenario, each dispatch center has vehicles and its own geographical operation area, and is responsible for deliveries initiated by certain factories. Since the operation areas may overlap, multiple dispatch centers may collaborate to handle a delivery. An agent representing a dispatch center can either announce a CNP to buy transport services from other dispatch centers, or bid in a CNP to sell its transport service. However, the assumptions in the scenario of [7] are different from those in our domain. For instance, no transport company in our travel domain should be responsible for the whole tourist track, since they only own some segments of the track. Moreover, since every visiting sequence of cities is possible, the dispatching agents will have the difficulty to initiate a CNP and specify which transport service to buy. Therefore it is difficult to directly apply the distributed route-forming approach in [7] to this problem.

7 Conclusion and Future Work

By applying multi-agent techniques to the travel domain, we found that the combination of existing graph theory and techniques, market oriented programming and agent negotiation can help to solve the travel plan optimization problems constrained by the balance between market efficiency and privacy. The question is how to integrate these techniques together without introducing side effects and problems. Currently we are implementing the strategies for the transportation agents who may involve in many negotiation processes at the same time. It is very interesting to see how transportation agents manage to deal with the negotiations in order to obtain the best profit according to their rationalities.

Acknowledgment. This research is supported by the MOE Program for Promoting Academic Excellence of Universities under grant number 89-E-FA04-1-4.

References

1. http://www.dot.gov/affairs/tsa13502.htm
2. Clemons, E., Hahn, I. and Hitt, L., 2001. Price dispersion and differentiation in on-line travel: an empirical investigation.
3. http://www.mapquest.com (Our system uses the maps drawn by this website)
4. U. Manber. *Introduction to Algorithms*. Addison-Wesley, Reading MA, 1989.
5. T.Cormen, C.Leiserson, and R.Rivest. *Introduction to Algorithms*. MIT Press, Cambridge MA, 1990.
6. W.E. Walsh, M.P. Wellman, P.R. Wurman, and J.K. MacKie-Mason. Auction protocols for decentralized scheduling. In Proceedings of the Eighteenth International Conference on Distributed Computing Systems (ICDCS-98), Amsterdam, the Netherlands, 1998.
7. Tuomas W Sandholm. An implementation of the Contract Net Protocol based on marginal-cost calculations. In Proc. of 11th National Conference on Artificial Intelligence (AAAI-93), pp. 256–262, July 1993.
8. Von-Wun Soo, Shu-How Liang, Recommendation a trip plan by negotiation, In Proc. of Cooperative Information Agents, 2001.

Agents for Intelligent Information Extraction by Using Domain Knowledge and Token-Based Morphological Patterns

Jaeyoung Yang and Joongmin Choi

Dept. of Computer Science and Engineering, Hanyang University
1271 Sa-1 Dong, Ansan-Si, Kyunggi-Do 426-791, Korea
{jyyang,jmchoi}@cse.hanyang.ac.kr

Abstract. Knowledge-based information extraction is known to have flexibility in recognizing various kinds of target information by exploiting the domain knowledge to automatically generate information-extraction rules. However, most of previous knowledge-based information-extraction systems are only applicable to labeled documents, and as a result, ontology terms must appear in the document in order to guide the system to determine the existence of the target information.
To make a knowledge-based information-extraction system to be more general enough to handle both labeled and unlabeled documents, this paper proposes an enhanced scheme of knowledge-based wrapper generation by using token-based morphological patterns. Each document is represented as a sequence of tokens, rather than a sequence of logical lines, in order to capture the meaning of data fragments more correctly and recognize the target information contextually. The newly implemented system XTROS$^+$ is presented and its performance is demonstrated.

1 Introduction

Information extraction(IE) is the task of recognizing and extracting specific data fragments from a collection of documents[5]. In general, the process of information extraction relies on a set of extraction rules, called a *wrapper*, tailored to a particular information source. Wrapper induction[4,6,7] has been suggested to automatically build a wrapper through learning from a set of resource's sample pages. Automatic wrapper induction can be based on either heuristics or domain knowledge.

Heuristic wrapper induction has been adopted by most traditional IE systems such as ARIADNE[1], SHOPBOT[3], STALKER[7], WHISK[9], and MORPHEUS[10]. An example heuristic that has been very useful is that *a text fragment with a dollar sign followed by a number (e.g., $250) can be regarded as the price information.* However, this approach is not effective since the heuristics are mostly simple and naive, and consequently, the system can extract only a limited number of apparent features such as the price. *Knowledge-based wrapper induction* tries to solve these problems by defining and applying the

J. Lee and M. Barley (Eds.): PRIMA 2003, LNAI 2891, pp. 74–85, 2003.

Fig. 1. Information extraction from labeled documents in XTROS

domain knowledge during wrapper generation. The knowledge-based approach is expected to extract more features such as the ISBN number of a book, the number of bedrooms in a house advertisement, and so on.

In previous efforts, we have built a prototype IE system named XTROS with the knowledge-based approach[11]. XTROS behaves effectively both in learning extraction rules and applying those rules to extract information from real estate pages sampled from the Web. Fig. 1 shows three *Home-for-Sale* Web pages obtained from popular real-estate sites and extraction results by XTROS. Fig. 2 is an XML-based domain knowledge for the domain of real estate.

XTROS proves that knowledge-based information extraction(KBIE) outperforms the heuristic methods in recognizing various kinds of target information. However, XTROS works correctly only on the labeled documents[2] in which the target information is always accompanied with a *label* or an *ontological term*. A label plays the role of a trigger that guides the system to examine the existence of the target data. For example, from the data fragment `author: John Brown`, the system can recognize that `John Brown` is the author of a book by checking the label `author:`. Unfortunately, the target information is often expressed without labels, and in this case, XTROS fails. Other KBIE systems have been facing with similar situations.

To resolve this problem, this paper proposes an improved scheme of knowledge-based wrapper generation by analyzing documents based on tokens and assigning morphological patterns to the target information. These morphological features contribute to enhance the recall measures even when the onto-

```
<KNOWLEDGE>
 <OBJECTS>
   <OBJECT>PRICE</OBJECT>
   <OBJECT>BED</OBJECT>
   <OBJECT>BATH</OBJECT>
   <OBJECT>CITY</OBJECT>
   <OBJECT>MLS</OBJECT>
   <OBJECT>DETAIL</OBJECT>
   <OBJECT>IMG</OBJECT>
 </OBJECTS>
 <PRICE>
   <ONTOLOGY>
     <TERM>PRICE</TERM>
     <TERM>$</TERM>
   </ONTOLOGY>
   <FORMAT>
     <FORM>[ONTOLOGY] DIGITS</FORM>
     <FORM>DIGITS [ONTOLOGY]</FORM>
   </FORMAT>
 </PRICE>
 <BED>
   <ONTOLOGY>
     <TERM>BEDROOMS</TERM>
     <TERM>BEDROOM</TERM>
     <TERM>BEDS</TERM>
     <TERM>BED</TERM>
     <TERM>BR</TERM>
     <TERM>BD</TERM>
   </ONTOLOGY>
   <FORMAT>
     <FORM>DIGITS [ONTOLOGY]</FORM>
     <FORM>[ONTOLOGY] DIGITS</FORM>
   </FORMAT>
 </BED>
 <BATH>
   <ONTOLOGY>
     <TERM>BATHROOMS</TERM>
     <TERM>BATHROOM</TERM>
     <TERM>BATHS</TERM>
     <TERM>BATH</TERM>
     <TERM>BA</TERM>
   </ONTOLOGY>
   <FORMAT>
     <FORM>DIGITS [ONTOLOGY]</FORM>
     <FORM>[ONTOLOGY] DIGITS</FORM>
   </FORMAT>
 </BATH>
 <MLS>
   <ONTOLOGY>
     <TERM>MLS ID:#</TERM>
     <TERM>MLS ID</TERM>
     <TERM>MLS#</TERM>
   </ONTOLOGY>
   <FORMAT>
     <FORM>[ONTOLOGY] DIGITS</FORM>
   </FORMAT>
 </MLS>
 <DETAIL>
     -- omitted ----
 </DETAIL>
 <IMG>
     -- omitted ----
 </IMG>
</KNOWLEDGE>
```

Fig. 2. XML representation of domain knowledge in XTROS

logical terms are missing in the documents. The newly implemented system is an extension of XTROS, and thereby named XTROS$^+$. In XTROS$^+$, each Web page is represented by a sequence of tokens, rather than a sequence of logical lines as in XTROS. Each token is associated with a tag that denotes its morphological nature. Still, ontology information is also exploited to handle the labeled documents efficiently. The domain knowledge of XTROS$^+$ includes both the ontological terms and the morphological patterns of the target information. As in XTROS, the domain knowledge and the wrappers are all represented in XML.

This paper is organized as follows. Section 2 presents the overall procedure for generating IE rules in XTROS$^+$. It also explains the preprocessing phase including the token generation. Section 3 describes the XML representation of the domain knowledge. Section 4 describes the wrapper-generation algorithm with the rule evaluation. Section 5 assesses the system with some experimental results. Finally, Section 6 concludes with the summary and future direction.

2 XTROS$^+$ System Overview

Overall procedure for the *wrapper generator* that produces IE rules is depicted in Fig. 3. The wrapper generator consults the domain knowledge to produce a wrapper through learning from training examples. In the preprocessing phase, training Web pages are converted into a sequence of tokens with morphological tags. In the rule-induction phase, candidate IE rules are produced by analyzing

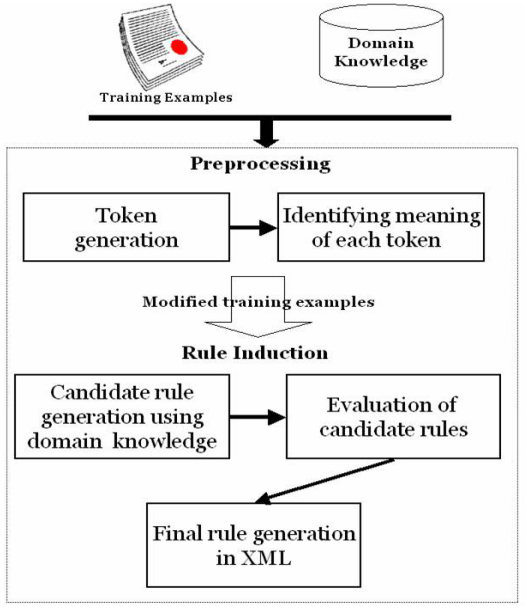


Fig. 3. Overall procedure for generating information-extraction rules

the tokens with the features in the domain knowledge. The generated rules are evaluated and represented in the XML form.

The preprocessing module transforms a Web document into a sequence of tokens. A *token* in XTROS$^+$ indicates a data fragment that is separable by a delimiter. For example, the sequence of tokens we would get from the HTML source of a Web document in Fig. 4(a), along with corresponding morphological tags, is shown in Fig. 4(b).

The fact that XTROS$^+$ treats a token as a basic unit of data analysis, rather than dividing the documents into logical lines, means that the document's structural characteristics such as sentential or phrasal features are ignored. At the first glance, the loss of structural information may seem to be weakening the extraction power of the system, but we claim that the correctness of information extraction depends not on these properties, but mainly on the position and the morphological pattern of the target information[8].

The concept of token in XTROS$^+$ has several characteristics. First, the process of recognizing and generating tokens is uniform and consistent regardless of the types of documents and the target information. Hence, a token-based IE system can be flexible enough to deal with both structured and unstructured documents. Second, a token, such as `CEO` or `PhD`, sometimes can be the target information in itself . This implies that token-based analysis may improve the recall measure when there are many meaningful tokens in a logical line. Third, the characters comprising a token can be categorized into alphabets, digits, and special characters. For example, a phone number `02 959-1234` contains two to-

<li class=small>Hardcover: 288 pages ; Dimensions (in inches): 0.88 x 9.68 x 6.30 <li class="small"> Publisher: Kluwer Academic Publishers; ISBN: 1402070616; (May 2002) <li class="small"> Amazon.com Sales Rank: 2,332,674

(a) HTML source of a Web document

Hardcover:	201	288	010	pages	100	;	001
Dimensions	200	(in	101	inches):	101	0.88	011
x	100	9.68	011	x	100	6.30	011
Publisher:	201	Kluwer	200	Academic	200	Publishers;	201
ISBN:	301	1402070616;	011	(May	201	2002)	011
Amazon.com	201	Sales	200	Rank:	201	2,332,674	011

(b) A sequence of tokens with morphological tags

Fig. 4. An example of token generation from a Web page

Table 1. Morphological tag assignments to tokens

Token	Morphological tag
$125.00	011
Friedemann	200
Agents:	201
affiliates	100
1-Click	211

kens `02`(with only digits) and `959-1234`(with digits and a special character `-`), and an email address `jyyang@cse.hanyang.ac.kr` is regarded as a single token with alphabets and a special character `@`.

Surprisingly, assigning a simple morphological tag to each token greatly improves the recognition rates of the meaning of tokens. A morphological tag assigned to each token is represented by a sequence of 3 digits. Table 1 shows an example of morphological tag assignments. The first digit in a tag provides the information about the alphabets the token contains: `0` means it does not contain any alphabets, `1` means all the alphabets are lowercase, `2` means only the first character is uppercase, and `3` means all are uppercase. The second digit indicates whether the token contains numerical digits, and the third digit indicates whether it contains special characters. For example, a morphological tag for the token `1-Click` is `211`, meaning that this token contains alphabets and only the first alphabet is uppercase(`C`), and it also contains a digit(`1`) and a special character (`-`). (More examples of tokens and their morphological tags can be found in Fig. 4(b).)

```
<KNOWLEDGE>
  <OBJECTS>
    <OBJECT> TITLE </OBJECT>
    <OBJECT> AUTHORS </OBJECT>
    <OBJECT> ISBN </OBJECT>
    ....
  </OBJECTS>
  ......
  <ISBN>
    <ONTOLOGY>
      <TERM> ISBN </TERM>
      <TERM> : </TERM>
    </ONTOLOGY>
    <SYNTACTICS>
      <MORPHOLOGY> 010 010 010 010 </MORPHOLOGY>
      <MORPHOLOGY> 010 </MORPHOLOGY>
      <MORPHOLOGY> 011 </MORPHOLOGY>
    </SYNTACTICS>
    <FORMAT>
      <FORM> [ONTOLOGY] [TARGET] </FORM>
    </FORMAT>
  </ISBN>
  ......
</KNOWLEDGE>
```

Fig. 5. Domain knowledge for ISBN

3 Extending Domain Knowledge with Morphological Patterns

Domain knowledge describes terms, concepts, and relationships between concepts widely used for a particular application domain. It plays a crucial role in recognizing the semantic fragments of a document in a given domain. We represent the domain knowledge using XML documents, and an example is shown in Fig. 5 that describes the ontologies and morphological patterns for recognizing the ISBN number of a book.

In our representation, the knowledge for a single domain is represented with the `<KNOWLEDGE>` element. The `<KNOWLEDGE>` construct contains an `<OBJECTS>` element that lists the target items to be extracted (called *objects*). The objects for the *book* domain may include TITLE, PRICE, AUTHORS, ISBN, PUBLISHER, and so on. An XML construct is maintained for each object. For example, the `<ISBN>` element corresponds to the ISBN object. Each XML construct for an object consists of three subelements: `<ONTOLOGY>`, `<SYNTACTICS>`, and `<FORMAT>`.

`<ONTOLOGY>` lists the terms that are used to recognize the existence of an object. The ISBN object has `ISBN` and `:` as its `<ONTOLOGY>` terms so that a fragment can be recognized as an ISBN number if it contains the string `"ISBN"` or the colon symbol.

The `<SYNTACTICS>` element describes the morphological characteristics of the target information. It lists potential morphological patterns that the target can have by using the `<MORPHOLOGY>` elements. Figure 5 shows that ISBN

```
Input: DOCS(Preprocessed Web Documents) and DK(Domain Knowledge)
Output: A wrapper containing IE rules in XML form

Algorithm WrapperInduction(){
  CRules[ ] ← LoadRulesFromKB(DK);
  FCP ← 0; // Frequency of Candidate Pattern
  for (i=0; i<the number of rules in CRules; i++){
    Rule ← CRules[i];
    FCP ← MatchMaking(Rule, DOCS);
    if (FCP>0){
      Register(Rule, FCP);
    }
  }
  TRules[ ] ← EvaluateRules(LoadRegisteredRules());
  FRules[ ] ← OptimizeRules(TRules);
  GenerateXMLWrapper(FRules);
}
```

Fig. 6. A pseudo-code for wrapper induction

can be in three different forms such as `030 2039 3039 0`, `030203930390`, or `030-2039-3039-0`. In the first case, 4 tokens contain only digits, so its morphological pattern should be `010 010 010 010` as shown in the figure. The second case is a single token with only digits, so its pattern should be `010`. Finally, the third case is also a single token with digits and special characters, so its pattern should be `011`. Note that the `<SYNTACTICS>` feature plays an important role in handling unlabeled documents.

`<FORMAT>` describes the positional relationship between the ontological terms and the target information. For example, the `<FORMAT>` of ISBN tells that the ISBN number appears after the ontological term. Even if there is no ontological term in `<FORMAT>`, XTROS$^+$ is able to extract the target information by considering the above-mentioned morphological patterns.

4 Knowledge-Based Wrapper Induction

Figure 6 shows a pseudo-code for wrapper induction algorithm. Inputs to this procedure are the domain knowledge and the preprocessed Web documents that are transformed into a sequence of tokens with morphological patterns.

The algorithm first consults the domain knowledge to generate all the possible situations in which the target information would occur. These candidate rules are temporarily stored in `CRules` array. Consider an example of extracting ISBN numbers from a Web document. According to the domain knowledge we discussed in the previous section, ISBN has three forms of morphological patterns. Therefore, there can be three candidate rules as follows:

ISBN [010 010 010 010]
ISBN [010]
ISBN [011]

Then, the system measures the support value for each candidate rule, as expressed by the **for** loop in the pseudo-code. The support value of a rule is increased whenever the ontology and the morphological pattern in the rule is matched with those in document tokens. Only those rules that are matched with at least one target information in the documents would be considered.

After finding the support values, rules are evaluated by unsupervised statistical approach. We define $\text{Freq}(R_i)$ as the matching frequency of a rule R_i, and n as the number of candidate rules. Only those candidate rules that satisfy the following equation are selected. Here, the choice of threshold value α can affect the recall measures. In general, α is set to a low value when we want to generate rules from a document where several patterns are repeated. Otherwise, α is set to a high value.

$$\frac{\text{Freq}(R_i)}{\sum_{i=1}^{n} \text{Freq}(R_i)} > \alpha, \qquad \text{where } (1 < i < n)$$

The selected rules are represented as a wrapper in XML. In XTROS$^+$, a wrapper is expressed by the `<Wrapper>` element. Fig. 7 is an example wrapper that contains the rules for recognizing the `<ISBN>` object. Each object consists of `<SEMANTICS>` and `<SYNTACTICS>` elements. `<SEMANTICS>` specifies an ontological term by using the `<ONTOLOGY>` element and indicates whether the term appears before or after the target value by using the `<STRUCTURE>` element. `<SYNTACTICS>` describes the morphological pattern for the target value with some constraints.

5 Experimental Results

The process of extracting the target information is done by the *wrapper interpreter* module. First, the Web documents are collected and preprocessed in the same way as done for the training examples. Then, the wrapper interpreter uses the value of the `<MORPHOLOGY>` element in the wrapper as a trigger to determine the existence of the target information.

The tokens that are matched with the morphological pattern in the wrapper become the candidate target information. Each candidate is examined to see if there is an ontological term before or after the token by consulting the value of the `<ONTOLOGY>` element. If the ontology exists, the `<STRUCTURE>` information is checked to see if the position of the term is matched with the rule. If there is no ontological terms, the morphological pattern is examined to extract the target information.

XTROS$^+$ is implemented in Java. Fig. 8 shows a screen shot of XTROS$^+$ execution for an Amazon search result page. It also shows the rule form that the interpreter is using with its ontological term (`Price`), the structure (`[ontology] [target]`), and the morphological pattern of the target (`011`).

```
<WRAPPER>
    ......
    <ISBN>
        <SEMANTICS ID="1">
            <ONTOLOGY> ISBN </ONTOLOGY>
            <STRUCTURE> [ONTOLOGY][TARGET] </STRUCTURE>
            <SYN> 1 </SYN>
        </SEMANTICS>

        <SEMANTICS ID="2">
            <ONTOLOGY> : </ONTOLOGY>
            <STRUCTURE> [ONTOLOGY][TARGET] </STRUCTURE>
            <SYN> 1 </SYN>
        </SEMANTICS>

        <SYNTACTICS ID="1">
            <MORPHOLOGY> 011 </MORPHOLOGY>
            <CONSTRAINT> SIZE 1 </CONSTRAINT>
        </SYNTACTICS>
    </ISBN>
    ......
</WRAPPER>
```

Fig. 7. An XML-based wrapper for extracting ISBN

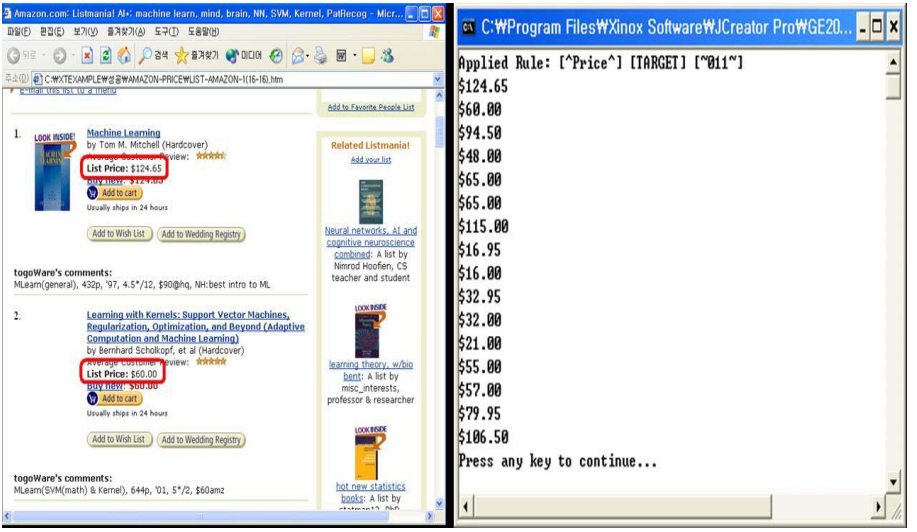
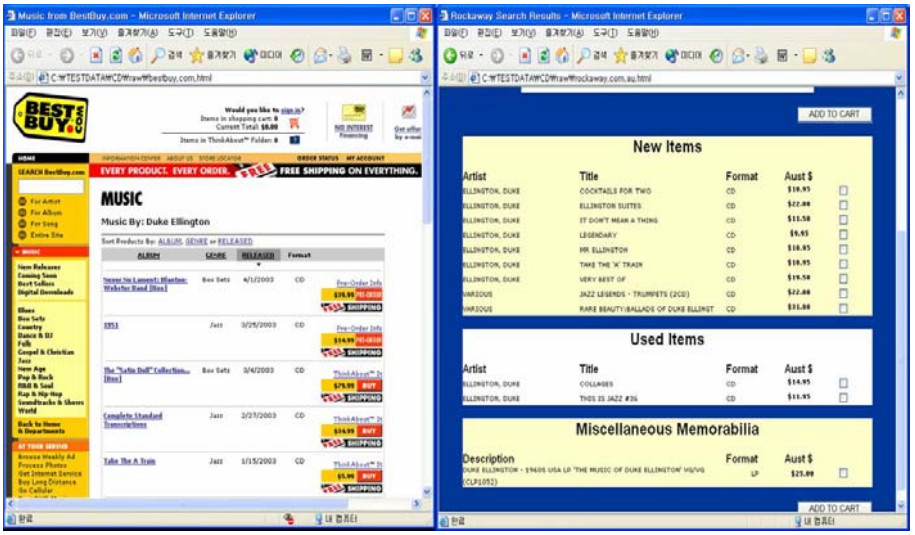


Fig. 8. A screen shot of extracting the PRICE information

XTROS$^+$ was run on the collection of Web pages that are obtained from the search results for some test queries in the book and CD ordering domain. To show the effectiveness of XTROS$^+$ which works for both labeled and unlabeled



(a) (b)

Fig. 9. Sample search results from BestBuy.com and Rockaway.com

documents, we have selected three different kinds of Web documents: 1) semi-structured documents with triggering labels or ontologies, such as Amazon query results in a list form as shown in Fig. 8, 2) structured documents with ontologies, such as BestBuy query results in a tabular form as shown in Fig. 9(a), and 3) structured documents without ontologies, such as Rockaway query results in a tabular form as shown in Fig. 9(b).

We have done a prelimenary analysis by examining the precision and recall measures for these 3 sites. For each site, we collected approximately 100 Web pages, each of which contains more or less 10 target items. Note that the threshhold value α is set to 50% in this experiment. Table 2 shows the result of this experiment. This table includes the data for the total number of target items to be extracted(T), the number of items actually extracted by XTROS$^+$(E), the number of items correctly extracted by XTROS$^+$ compared to the target items(C), the precision measure(P), and the recall measure(R). The precision is calculated by the formula $P = \frac{C}{E}$, and the recall is calculated by $R = \frac{C}{T}$.

As revealed in the table, XTROS$^+$ achieved high recall measures for most sites, regardless of the existence of labels in the documents. This resulted from the token-based analysis of the documents and the abstract representation of tokens by using morphological features. However, this may sometimes affect the precision measures negatively, as revealed in the precision value of BestBuy.com. We think that this might be related to the value of α. We are currently working on finding the reasonable threshold values with a series of extensive experiments.

Table 2. An experimental result

Sample Sites	Target Info.	Ontology	# of target items(T)	# of extracted items(E)	# of correctly extracted items(C)	Recall (R) R=C/T	Precision (P) P=C/E
Amazon(Single)	ISBN	ISBN	100	100	100	100%	100%
Amazon(List)	PRICE	PRICE	1200	1464	1200	100%	82 %
Amazon(List)	TITLE	BY	1260	1327	1234	98%	95%
Rockaway	PRICE	N/A	1752	1947	1752	100%	90%
BestBuy	PRICE	N/A	850	1308	850	100%	65%

Amazon URL: www.amazon.com
(Amazon(Single) indicates a detailed description page for a single item)
Rockaway URL: www.rockaway.com
BestBuy URL: www.bestbuy.com

6 Conclusion

We have presented an improved scheme of knowledge-based wrapper generation by analyzing documents based on tokens with morphological patterns. Our implemented system, XTROS$^+$, is able to handle both labeled and unlabeled documents by using the domain knowledge and token-based morphological patterns. XTROS$^+$ shows good performance on several Web sites in the domain of book ordering, and it is expected to be adaptable to different domains with easy modification of the XML-based domain knowledge.

In the future, we need to do more experiments for extensive analysis of the system performance. We also need to cope with the situations where the morphological patterns are the same but the natures of the data fragments are different. We are currently tackling it by investigating a method of building more semantical ontologies.

References

1. J. Ambite, N. Ashish, G. Barish, C. Knoblock, S. Minton, P. Modi, I. Muslea, A. Philpot, S. Tejada: ARIADNE: A system for constructing mediators for Internet sources. In: L.M. Haas, A. Tiwary (eds.), *Proc. ACM SIGMOD Int. Conf. on Management of Data, Seattle, WA, 2–4 June 1998* (ACM Press 1998) pp. 561–563
2. A. Blum, T. Mitchell: Combining labeled and unlabeled data with co-training. In: *Proc. 11th Conf. on Computational Learning Theory, Madison, WI, 24–26 July 1998* (ACM Press 1998) pp. 92–100
3. R. Doorenbos, O. Etzioni, D. Weld: A scalable comparison-shopping agent for the World Wide Web. In: *Proc. 1st Int. Conf. on Autonomous Agents, Marina del Rey, CA, 5–8 Feburary 1997* (ACM Press, New York 1997) pp. 39–48
4. N. Kushmerick, D. Weld, R. Doorenbos: Wrapper induction for information extraction. In: *Proc. 15th Int. Joint Conf. on Artif. Intell., Nagoya, Japan, 23–29 August 1997* (Morgan Kaufmann 1997) pp. 729–735

5. N. Kushmerick: Gleaning the Web. IEEE Intelligent Systems 14, 20–22 (1999)
6. N. Kushmerick: Wrapper induction: efficiency and expressiveness. Artif. Intell. 118, 15–68 (2000)
7. I. Muslea, S. Minton, C. Knoblock: A hierarchical approach to wrapper induction. In: *Proc. 3rd Int. Conf. on Autonomous Agents, Seattle, WA, 1–5 May 1999* (ACM Press 1999) pp. 190–197
8. E. Riloff: Automatically constructing a dictionary for information extraction tasks. In: *Proc. 11th Nat. Conf. on Artif. Intell., Washington, DC, 11–15 July 1993* (AAAI Press/The MIT Press 1993) pp. 811–816
9. S. Soderland: Learning information extraction rules for semi-structured and free text. Machine Learning 34, 233–272 (1999)
10. J. Yang, E. Lee, J. Choi : A shopping agent that automatically constructs wrappers for semi-structured online vendors. In: K. Leung, L. Chan, H. Meng (eds.), *Intelligent Data Engineering and Automated Learning*, Lecture Notes in Computer Science, vol. 1983 (Springer, Berlin, Heidelberg, New York 2000) pp. 368–373
11. J. Yang, J. Choi: Knowledge-based wrapper induction for intelligent Web information extraction. In: N. Zhong, J. Liu, Y. Yao (eds.), *Web Intelligence* (Springer 2003).

Using Web Usage Mining and SVD to Improve E-commerce Recommendation Quality

Jae Kyeong Kim[1] and Yoon Ho Cho[2]

[1] School of Business Administration, KyungHee University
#1, Hoeki-dong, Dongdaemoon-gu, Seoul, 130-701, Korea
Tel: +82-2-961-9355, Fax: +82-2-967-0788
jaek@khu.ac.kr
[2] Department of Internet Information, Dongyang Technical College
62-160, Gochuck-dong, Guro-gu, Seoul, 152-714, Korea
Tel: +82-2-2610-1918, Fax: +82-2-2610-1859
yhcho@dongyang.ac.kr

Abstract. Collaborative filtering is the most successful recommendation method, but its widespread use has exposed some well-known limitations, such as sparsity and scalability. This paper proposes a recommendation methodology based on Web usage mining and SVD (Singular Value Decomposition) to enhance the recommendation quality and the system performance of current collaborative filtering-based recommender systems. Web usage mining populates the rating database by tracking customers' shopping behaviors on the Web, so leading to better quality recommendations. SVD is used to improve the performance of searching for nearest neighbors through dimensionality reduction of the rating database. Several experiments on real Web retailer data show that the proposed methodology provides higher quality recommendations and better performance than other recommendation methodologies.

1 Introduction

The movement toward e-commerce has allowed companies to provide customers with more choices on products. Increasing choice has also caused product overload where the customer is no longer able to effectively choose the products he/she is exposed to. A promising technology to overcome the product overload problem is recommender systems that help customers find the products they would like to purchase. To date, a variety of recommendation techniques have been developed. Collaborative Filtering (CF) is the most successful recommendation technique, which has been used in a number of different applications such as recommending movies, articles, products, Web pages, etc. [2], [6], [13], [16].

CF-based recommender systems recommend products to a target customer according to the following steps [15]: (1) A customer provides the system with preference ratings of products that may be used to build a customer profile of his or her likes and dislikes. (2) Then, these systems apply statistical techniques or machine learning techniques to find a set of customers, known as *neighbors*,

J. Lee and M. Barley (Eds.): PRIMA 2003, LNAI 2891, pp. 86–97, 2003.

which in the past have exhibited similar behavior (i.e., they either rated similarly or purchased similar set of products). Usually, a neighborhood is formed by the degree of similarity between the customers. (3) Once a neighborhood of similar customers is formed, these systems predict whether the target customer will like a particular product by calculating a weighted composite of the neighbors' ratings of that product (*prediction problem*), or generate a set of products that the target customer is most likely to purchase by analyzing the products the neighbors purchased (*top-N recommendation problem*). These systems, also known as the nearest neighbor CF-based recommender systems have been widely used in practice. However, as the number of customers and that of products managed in an e-commerce site grow rapidly, its application to e-commerce has exposed two major issues that must be addressed [3], [15].

The first issue is related to sparsity. In a large e-commerce site such as Amazon.com, there are millions of products and so customers may rate only a very small portion of those products. Most similarity measures used in CF work properly only when there exists an acceptable level of ratings across customers in common. Such a sparsity in ratings makes the formation of neighborhood inaccurate, thereby results in poor recommendation. The second issue is related to scalability. Recommender systems for large e-commerce sites have to deal with millions of customers and products. Because these systems usually handle very high-dimensional profiles to form the neighborhood, the nearest neighbor algorithm is often very time-consuming and scales poorly in practice.

In this paper, we propose a recommendation methodology, called Web usage mining driven Collaborative Filtering recommendation methodology using SVD (WebCF-SVD), to address the sparsity and scalability problems of current CF-based recommender systems.

Web usage mining is employed to address the sparsity problem. Web usage mining analyzes customers' shopping behaviors on the Web and collects their implicit ratings. This increases the number of ratings rather than only collecting explicit ratings, thereby reduces the sparsity. E-commerce data are rich and detailed compared to off-line commerce data. One type of collected e-commerce data is a clickstream that tracks visitors' path through a Web site. The clickstream in Web retailers provides information essential to understanding the shopping patterns or prepurchase behaviors of customers such as what products they see, what products they add to the shopping cart, and what products they buy. By analyzing such information via Web usage mining, it is possible not only to make a more accurate analysis of the customer's interest or preference across all products (than analyzing the purchase records only), but also to increase the number of ratings (when compared to collecting explicit ratings only). Nevertheless, the existing research in recommender systems has not offered a formal way for capturing implicit ratings of individual customer through Web usage mining. In this paper, we suggest a formal scheme to capture implicit ratings by analyzing customers' online shopping behaviors and to build the customer profiles. To solve the scalability problem, we use a nearest neighbor CF algorithm. But, before applying the algorithm, we reduce the dimensionality of

the customer profiles. As a dimensionality reduction technique, we employ singular value decomposition (SVD) which concentrates most of the information in a few dimensions. SVD is a matrix factorization technique used for producing low-rank approximation of the original space. It enables us to search for the neighbors using the low dimensional and dense customer profile. The dimensionality reduction using SVD thus leads to reducing the sparsity of preference ratings as well as improving the scalability of making recommendations.

2 Background

2.1 Web Usage Mining

Web usage mining is the process of applying data mining techniques to the discovery of behavior patterns based on Web log data, for various applications. In the advance of e-commerce, the importance of Web usage mining grows larger than before. The overall process of Web usage mining is generally divided into two main tasks: data preprocessing and pattern discovery. Mining behavior patterns from Web log data needs the data preprocessing tasks that include data cleansing, user identification, session identification, and path completion. More detailed information for data preprocessing of the Web log can be found in [4]. The pattern discovery tasks involve the discovery of association rules, sequential patterns, usage clusters, page clusters, user classifications or any other pattern discovery method [12]. The usage patterns extracted from Web data can be applied to a wide range of applications such as Web personalization, system improvement, site modification, business intelligence discovery, usage characterization, etc. [17].

There have been several customer behavior models for e-commerce, which have different analysis purposes. Menascé et al. [11] presented a state transition graph, called Customer Behavior Model Graph (CBMG) to describe the behavior of groups of customers who exhibit similar navigational patterns. VandeMeer et al. [18] developed a user navigation model designed for supporting and tracking dynamic user behavior in online personalization. Lee et al. [9] provided a detailed case study of clickstream analysis from an online retail store. To measure the effectiveness of efforts in merchandizing, they analyzed the shopping behavior of customers according to the following four shopping steps: product impression, click-through, basket placement, and purchase.

2.2 SVD

SVD has been used to improve performance in information retrieval or information filtering. It is a matrix factorization technique commonly used for projecting high dimensional (sparse) data into a low dimensional (dense) space.

The SVD of $m \times n$ matrix $\mathbf{P}$ with rank r is its factorization into a product of three matrices.

$$\mathbf{P} = \mathbf{USV}^T \tag{1}$$

where $\mathbf{U}$ and $\mathbf{V}$ are orthogonal matrices and the diagonal entries of $\mathbf{S}$ are the singular values of $\mathbf{P}$ sorted in descending order. $\mathbf{U}$ is a $m \times r$ matrix and implies the preference of each customer to certain features. $\mathbf{V}$ is a $n \times r$ matrix and denotes the amount of each feature included in each product. $\mathbf{S}$ is a $r \times r$ matrix and indicates the importance of the feature. It is possible to reduce dimensionality by choosing the k most significant dimensions from the factor space which is then used for estimating the original vectors. If we keep k dimensions, matrix $\mathbf{P}_k$=$\mathbf{U}_k\mathbf{S}_k\mathbf{V}_k^T$ is the rank-k approximation of the original matrix $\mathbf{P}$, $\mathbf{U}_k$, $\mathbf{S}_k$ and $\mathbf{V}_k$ is a $m \times k$, $k \times k$, $n \times k$ matrix, respectively.

Many researchers have emphasized the usage of SVD as dimensionality reduction technique. In particular, Kanth et al. [7] showed that the low dimensional data obtained to use the SVD support fast searching because overhead of computation is reduced.

3 Methodology

WebCF-SVD is a recommendation methodology using Web usage mining and SVD to improve the recommendation quality and the system performance of current CF-based recommender systems. The entire procedure of WebCF-SVD is divided into four phases: customer profile creation, dimensionality reduction, neighborhood formation, and recommendation generation.

3.1 Phase 1: Customer Profile Creation

A customer profile is a description of customer interests or preferences about products. In CF, recommending products to a particular customer depends on his/her profile. This phase discovers customer's preferences across all the products, and makes the customer profile based on the preferences. For this purpose, the customer profile is constructed based on the following three general shopping steps in Web retailers:

- *click-through* : the click on the hyperlink and the view of the Web page of the product
- *basket placement* : the placement of the product in the shopping basket,
- *purchase* : the purchase of the product.

A basic idea of measuring the customer's preference is simple and straightforward. The customer's preference is measured by counting the number of occurrence of URLs mapped to the product from clickstream of the customer. In Web retailers, products are purchased in accordance with the three sequential shopping steps: click-through, basket placement, and purchase. Hence, we can classify all products into four product groups such as purchased products, products placed in the basket, products clicked through, and the other products. This classification provides an *is-a* relation between different groups such that purchased products *is-a* products placed in the basket, and products placed in the basket *is-a* products clicked through. From this relation, it is reasonable to

obtain a preference order between products such that {products never clicked} $\prec$ {products only clicked through } $\prec$ {products only placed in the basket} $\prec$ {purchased products}. Hence, it makes sense to assign the higher weight to occurrences of purchased products than those of products only placed in the basket. Similarly, the higher weight is given to products only placed in the basket than those of products only clicked through, and so on.

Let p_{ij}^c, p_{ij}^b and p_{ij}^p be the total number of occurrences of click-throughs, basket placements and purchases of a customer i for a product j, respectively. They are calculated from the raw clickstream data as the sum over the given time period, and so reflect individual customer's behaviors in the corresponding shopping process over multiple shopping visits.

From the above notation, we define the customer profile as the matrix of ratings $\mathbf{P} = (p_{ij}), i = 1, \cdots, m$ (total number of customers), $j = 1, \cdots, m$ (total number of products), where

$$p_{ij} = \begin{cases} \dfrac{p_{ij}^c}{\sum_{i=1}^{m} p_{ij}^c} + \dfrac{p_{ij}^b}{\sum_{i=1}^{m} p_{ij}^b} + \dfrac{p_{ij}^p}{\sum_{i=1}^{m} p_{ij}^p} & \text{if } p_{ij}^c > 0 \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

p_{ij} is a sum of the normalized value of p_{ij}^c, p_{ij}^b and p_{ij}^p. It ranges from 0 to 3, where more preferred product results in bigger value. Note that the weights for each shopping step are not the same although they look equal as in Equation (2). From a casual fact that customers who purchased a specific product had already not only clicked several Web pages related to it but placed it in the shopping basket, we can see that Equation (2) reflects the different weights.

3.2 Phase 2: Dimensionality Reduction

This phase factors the $m \times n$ matrix $\mathbf{P}$ and obtains a rank-k ($k \ll n$) approximation of the original matrix. Using only k reduced product dimension, the neighborhood formation task becomes more scalable. The optimal value of k is usually determined by performing experiments with several different values [7], [14]. The dimensionality reduction is performed according to the following steps [14]:

Step 1. Calculate the SVD of $\mathbf{P}$ and obtain the matrices $\mathbf{U}$, $\mathbf{S}$ and $\mathbf{V}$ of dimension $m \times m$, $m \times n$, and $n \times n$, respectively.

Step 2. Obtain a $k \times k$ matrix $\mathbf{S}_k$ by keeping only k diagonal entries of the matrix $\mathbf{S}$. Similarly, obtain matrices $\mathbf{U}_k$ and $\mathbf{V}_k$ of dimension and $m \times k$, $k \times n$, respectively.

Step 3. Calculate the matrix product $\mathbf{U}_k \cdot \sqrt{\mathbf{S}}_k$. The $m \times k$ matrix $\mathbf{U}_k \cdot \sqrt{\mathbf{S}}_k$ represents that all n customers now provide their ratings on the k *meta*-products, because it is a rank-k approximation of the original matrix of ratings $\mathbf{P}$.

3.3 Phase 3: Neighborhood Formation

This phase performs computing the similarity between customers and, based on that, forming a neighborhood between a target customer and a number of like-minded customers. The process follows the same manner as that of typical nearest-neighbor algorithms except forming the neighborhood in reduced dimensional space. The details of the neighborhood formation are as follows.

The similarity between two customers a and b is measured by calculating the cosine of the angle between the two row vectors in the k-dimensional space, $\cos(\overrightarrow{a}, \overrightarrow{b})$, which is defined as follows:

$$\cos(\overrightarrow{a}, \overrightarrow{b}) = \frac{\overrightarrow{a} \cdot \overrightarrow{b}}{\|\overrightarrow{a}\| \|\overrightarrow{b}\|} \tag{3}$$

Using the cosine measure, this phase determines which customers are used in the recommendation for the target customer. Two well-known techniques, *correlation-thresholding* and *best-n-neighbors*, are generally used to determine how many neighbors to select [5]. Correlation-thresholding technique is to form a neighborhood as customers with absolute correlates greater than a given threshold, while best-n-neighbors technique is to select the best correlates for the neighbors. In WebCF-SVD, the best-n-neighbors technique is adopted because of its superiority in performance over the other [5].

3.4 Phase 4: Recommendation Generation

The final phase of WebCF-SVD is to ultimately derive the *top-N* recommendation from the neighborhood of customers. For each customer, we produce a recommendation list of products that the target customer is most likely to purchase. Previously purchased products are excluded from the recommendation list in order to broaden each customer's purchase patterns or coverage.

To generate a recommendation list for a given customer, WebCF-SVD looks into the neighborhood and for each neighbor, scans through clickstream data and counts the *reference frequency* of the products. After all neighbors are accounted for, the system sorts the products according to their frequency count and returns N most frequently referred products as the recommendation list. The reference frequency of the neighborhood of a particular customer a for a product j, $RF_{a,j}$, is defined below:

$$RF_{a,j} = \sum_{i \in \text{neighbors of customer a}} \left(\frac{r_{ij}^c}{\sum_{j=1}^{n} r_{ij}^c} + \frac{r_{ij}^b}{\sum_{j=1}^{n} r_{ij}^b} + \frac{r_{ij}^p}{\sum_{j=1}^{n} r_{ij}^p} \right) \times \cos(\overrightarrow{a}, \overrightarrow{i}) \tag{4}$$

where n is the number of products, and r_{ij}^c, r_{ij}^b and r_{ij}^p is the total number of occurrences of click-throughs, basket placements and purchases of a customer i for a product j, respectively, and $\cos(\overrightarrow{a}, \overrightarrow{i})$ is the similarity between two customer a and i. Our method follows from the hypothesis that the more a product is referred, the higher the possibility of product's purchase becomes.

4 Experimental Evaluation

4.1 Data Sets

For our experiments, we used Web log data and product data from C Web retailer in Korea that sells a variety of beauty products. One hundred and twenty-four Web log files were collected from four IIS Web servers during the period between May 1, 2001 and May 30, 2001. The total size of the Web log files was about 64,730MB, and total number of HTTP requests was about 420,000,000,000. For application to our experiments, data preprocessing tasks such as data cleansing, user identification, session identification, and path completion were applied to the Web log files. Finally, we obtained a transaction database in the form of <time, customer-id, product-id, shopping-step> where the shopping-step is the click-through step, the basket-placement step or the purchase step. This database contains the transactions of 66,329 customers. In total, the database contains 2,249,540 records that consist of 7,208 purchase records, 60,892 basket-placement records, and 2,181,440 click-through records.

The period between May 1, 2001 and May 24, 2001 was set as training period and the period between May 25, 2001 and May 30, 2001 was set as test period. As the target customers, we selected 116 customers who have purchased at least one product in both periods. Finally, the training set consisted of 8,960 transaction records created by the target customers for the training period, and the test set consisted of 156 purchase records created by them for the test period.

4.2 Evaluation Metrics

The existing studies about recommender systems have used a number of different measures for evaluating the success of a recommender system. The main objective of this research is to develop an effective and efficient recommendation methodology that has better quality and less computation time compared to other methodologies. This research employs two evaluation metrics for evaluating our methodology in terms of quality and performance requirements.

Quality Evaluation Metric. With the training set and the test set, WebCF-SVD works on the training set first, and then it generates a set of recommended products, called recommendation set, for a given customer. To evaluate the quality of the recommendation set, recall and precision have been widely used in recommender systems research [1], [10], [14]. Recall is defined as the ratio of the number of products in both the test set and the recommendation set to the number of products in the test set. Precision is defined as the ratio of the number of products in both the test set and the recommendation set to the number of products in the recommendation set. Recall means how many of all the products in the actual customer purchase list are recommended correctly whereas precision means how many of the recommended products belong to the actual customer purchase list. These measures are simple to compute and intuitively

appealing, but they are in conflict since increasing the size of the recommendation set leads to an increase in recall but at the same time a decrease in precision [15]. Hence, a widely used combination metric called $F1$ metric [1], [8], [15] that gives equal weight to both recall and precision was employed for our evaluation. It is computed as follows:

$$F1 = \frac{2 \times recall \times precision}{recall + precision} \tag{5}$$

Performance Evaluation Metric. To evaluate the scalability issue, we used a performance evaluation metric in addition to the quality evaluation metric. The *response time* was employed to measure the system performance. The response time defines the amount of time required to compute all the recommendations for the training set.

Experimental Environments and Benchmark Recommender Systems. A system to perform our experiments was implemented using Visual Basic 6.0 and ADO components. The system consists of two parts - one for Web log preprocessing and the other for experiment execution and result To evaluate the scalability issue, we used a performance evaluation metric in addition to the quality evaluation metric. The response time was employed to measure the system performance. The response time defines the amount of time required to compute all the recommendations for the training set. analysis. MS-Access is used to store and process all the data necessary for our experiments. We run our experiments on Windows 2000 based PC with Intel Pentium III processor having a speed 750 MHz and 1GB of RAM. To verify the effectiveness of WebCF-SVD, we compare our results to those of benchmark recommender systems. For this purpose, two benchmark recommender systems were also implemented in Visual Basic 6.0. One is a SVD-based recommender system (CF-SVD) that applies SVD to reduce the dimensionality of a customer profile derived from not Web-usage data but purchase data only. CF-SVD represents the customer profile, $\mathbf{P}$, such that p_{ij} is one if a customer i has purchased a product j, and zero, otherwise. The other is a typical CF-based recommender system (Pure-CF) that builds a customer profile using purchase data in the same manner as that of CF-SVD, but does not apply SVD to recommendation process.

4.3 Experiment Results and Discussions

This section presents a detailed experimental evaluation of the different parameters for the phases of WebCF-SVD and compares their performance to those of the benchmark CF algorithms. As the combination of different parameters is enormous, we first determine the optimal values of different parameters, and then use them for the rest of the experiments. Main experimental parameters of WebCF-SVD considered during our experiments were as follows:

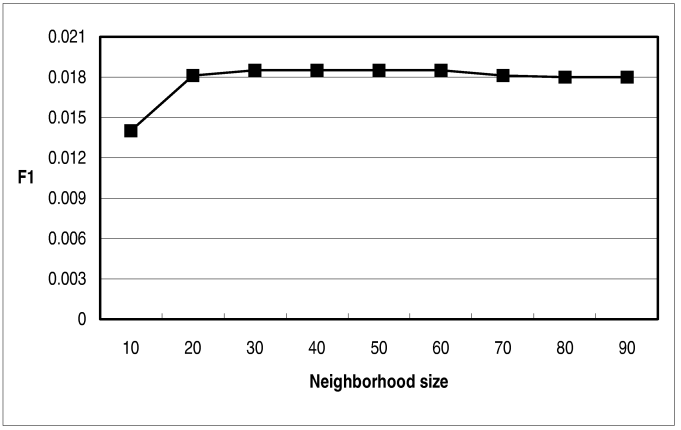


Fig. 1. Impact of neighborhood size on recommendation quality

- Size of neighborhood
- Number of dimension
- Number of recommended products

Impact of Neighborhood Size. The size of the neighborhood is reported to have a significant impact on the recommendation quality [5]. To determine the sensitivity of neighborhood size, we performed an experiment where we varied the number of neighbors and computed the corresponding $F1$ metric. Fig. 1 shows our experimental results. Looking into the results, we can conclude that the size of the neighborhood does affect the quality of *top-N* recommendations. Generally, the recommendation quality increases as the number of neighbors increases, but, after a certain peak, the improvement gains diminish and the quality becomes worse [15]. Choosing too many neighbors is thought to result in too much noise for those who have high correlates. In our experiment, the peak was reached at the area between 40 and 60. We thus used 50 as our choice of neighborhood size.

Impact with Number of Dimension. When using SVD as a dimensionality reduction methodology, the number of dimension has an impact on the recommendation quality. That is, the number of dimension (k) is critical for the effectiveness of the low dimensional representation. To avoid over-fitting error, we performed the experiments with diverse number of dimensions that is large enough. This result is shown in Fig. 2. The recommendation quality reaches its peak quickly in the rank of lower dimensional space. For the rest of the experiments, we fixed k at 11.

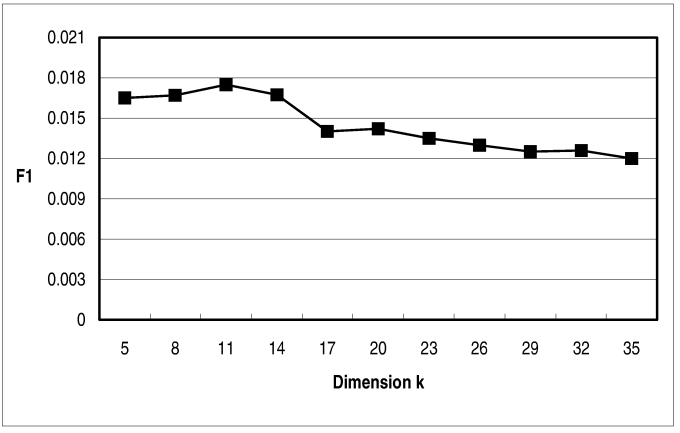


Fig. 2. Impact with the dimension k

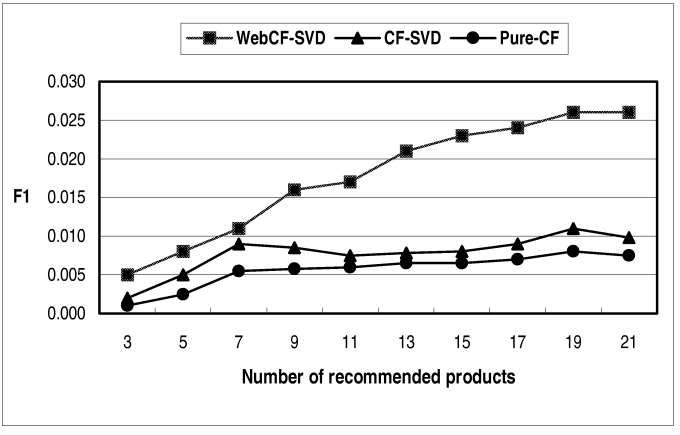


Fig. 3. Quality comparison of WebCF-PT and benchmark CF algorithms

Quality Comparison with the Benchmark CF Algorithm. Once we obtained the optimal values of the parameters, we compared the recommendation quality of WebCF-SVD with those of the benchmark CF algorithms. Fig. 3 shows our experimental results. It can be observed from Fig. 3 that WebCF-SVD works better than the benchmark CF algorithms at all the number of recommended products. Compared to CF-SVD and Pure-CF, WebCF-SVD achieves an average improvement of 128% and 214%, respectively. We can reach the following conclusions through the result of experiments. First, the performance of CF using the Web data is better than that of CF using the purchase data only. Second, the recommendation qualities of CF algorithms using dimensionality reduction technique are better than those of CF using original sparse rating data.

Table 1. Performance comparison of WebCF-PT and benchmark CF algorithms

	WebCF-SVD	CF-SVD	Pure-CF
Response Time (sec.)	22.5	13.2	263.4

Performance Comparison with the Benchmark CF Algorithm. To compare the performance of WebCF-SVD with those of the benchmark CF algorithms, we performed an experiment to measure the response time of each algorithm. Table 1 shows the response time (seconds) by the three algorithms. Looking into the results shown in Table 1, we can see that the scalability-related performances of CF algorisms using dimensionality reduction are about ten times better than that of CF using high dimensional data. This point is very important in the e-commerce environment because the number of products and that of customers grow very fast.

5 Conclusion

The rapid expansion of e-commerce forces existing recommender systems to deal with a large number of customers and products and to ensure high quality of recommendations. In this paper, we focused on these challenging issues of the recommender systems and proposed a recommendation methodology where we apply Web usage mining and SVD to address these issues together. Experimental results show that the methodology works better and faster than existing CF-based recommender systems in an e-commerce environment.

The research work presented in this paper makes several contributions to the recommender systems related research. First, we applied SVD both to reducing the sparsity in the rating database and to improving the scalability of searching for neighbors. Second, we developed a Web usage mining technique to capture implicit ratings by tracking customers' shopping behaviors on the Web and applied it to reducing the sparsity. Third, we developed a Web usage mining technique to choose proper products to recommend from the neighborhood.

While our experimental results suggest that the proposed methodology is effective and efficient for product recommendations in the e-commerce environment, these results are based on data sets limited to the particular e-commerce site. Therefore, it is required to evaluate our methodology in more detail using data sets from a variety of large e-commerce sites. Furthermore, it will be an interesting research area to conduct a real marketing campaign to target customers using our methodology and then to evaluate its performance.

References

1. Billsus, D., Pazzani, M. J.: Learning collaborative information filters. In Proc. 15th International Conference on Machine Learning. (1998) 46–54.
2. Cho, Y. H., Kim, J. K., Kim, S. H.: A personalized recommender system based on Web usage mining and decision tree induction. Expert Systems with Applications. 23 (2002) 329–342.
3. Claypool, M., Gokhale, A., Miranda, T., Murnikov, P., Netes, D., Sartin, M.: Combining content-based and collaborative filters in an online newspaper. In Proc. ACM SIGIR'99 Workshop on Recommender Systems, Berkeley, CA. (1999).
4. Cooley, R., Mobasher, B., Srivastava, J.: Data preparation for mining world wide web browsing patterns. Journal of Knowledge and Information Systems. 1 (1999) 5–32.
5. Herlocker, J., Konstan, J., Borchers, A., Riedl, J.: An algorithmic framework for performing collaborative filtering. In Proc. Conference on Research and Development in Information Retrieval. (1999) 230–237.
6. Hill, W., Stead, L., Rosenstein, M., Furnas, G.: Recommending and evaluating choices in a virtual community of use. In Proc. ACM Conference on Human Factors in Computing Systems. 194–201.
7. Kanth, K. V., Agrawal, D., Abbadi, A. E., Singh, A.: Dimensionality Reduction for Similarity Searching in Dynamic Databases, Computer Vision and Image Understanding. 75 (1999) 59-72.
8. Kim, J. K., Cho, Y. H., Kim, W. J., Kim, J. R., Suh, J. Y.: A personalized recommendation procedure for Internet shopping support. Electronic Commerce Research and Applications. 1 (2002) 301–313.
9. Lee, J., Podlaseck, M., Schonberg, E., Hoch, R.: Visualization and analysis of clickstream data of online stores for understanding Web merchandising. Data Mining and Knowledge Discovery. 5 (2001) 59–84.
10. Lin, W., Alvarez, S. A., Ruiz, C.: Efficient adaptive-support association rule mining for recommender systems. Data Mining and Knowledge Discovery. 6 (2001) 83–105.
11. Menasce, D. A., Almeida, V. A., Fonseca, R., Mendes, M. A.: A methodology for workload characterization of e-commerce sites. In Proc. ACM E-Commerce. (1999) 119–128.
12. Mobasher, B., Cooley, R., Srivastava, J.: Automatic personalization based on Web usage mining. Communications of the ACM. 43 (2000) 142–151.
13. Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., Riedl, J.: GroupLens an open architecture for collaborative filtering of netnews. In Proc. ACM 1994 Conference on Computer Supported Cooperative Work. (1994) 175–186.
14. Sarwar, B., Karypis, G., Konstan, J. A., Riedl, J.: Application of dimensionality reduction in recommender system – a case study. In Proc. ACM WebKDD-2000 Workshop. (2000).
15. Sarwar, B., Karypis, G., Konstan, J. A., Riedl, J.: Analysis of recommendation algorithms for e-commerce. In Proc. ACM E-Commerce. (2000) 158–167.
16. Shardanand U. Maes, P.: Social information filtering algorithms for automating "word of mouth". In Proc. Conference on Human factors in Computing Systems. (2000) 210–217.
17. Srivastava, J., Cooley, R., Deshpande, M., Tan, P.: Web usage mining: discovery and applications of usage patterns from Web data. SIGKDD Explorations. 1 (2000) 1–12.
18. VanderMeer, D., Dutta, K., Datta, A.: Enabling scalable online personalization on the Web. In Proc. ACM E-Commerce. (2000) 185–196.

Semantic Web Service Architecture Using Multi-agent Scenario Description

Sachiyo Arai, Yohei Murakami, Yuki Sugimoto, and Toru Ishida

Department of Social Informatics, Kyoto University,
606-8501 Kyoto, Japan
{sachiyo,yohei,sugimoto,ishida}@kuis.kyoto-u.ac.jp

Abstract. Current research issues on web services have come to center around flexible composition of existing services. Under the initiative of industry, flexible composition framework has been developed on a workflow model where flow of the processes and bindings among services should be known beforehand. In short, its framework realizes flexible composition within the available services that are not widely opened. This paper focuses on two limitations of current web service composition. One limitation is that it's hard to represent multi-agent scenarios consisting of several concurrent processes because it is based on a workflow model. The other limitation is that once composed, web service cannot be reused or transferrd for other requesters, because there is no function to put the semantics on composite web service. To overcome these limitations, we have developed scenario description language *Q*, which enables us to realize a web service composition reflecting a multi-agent's context not as a workflow but a scenario. Furthermore, we developed a system that translates multi-agent scenario to DAML-S, and that registers the translated DAML-S as a new Web service. We also discuss the availability of our system for designing an application to C-Commerce and Digital Cities.

1 Introduction

Web technologies, such as Universal Description, Discovery, and integration (UDDI) [UDDI 2002], Web Services Description Language (WSDL) [Christensen 01] and Simple Object Access Protocol (SOAP) [Box 00] have extended people's usage of information on the Web. The most recent technologies, such as Business Process Execution Language for Web Services (BPEL4WS)[Virdell 03] and Web Service Choreography Interface (WSCI)[Arkin et.al 01] realize a dynamic composition of web services. However, there remain some difficulties within these frameworks to realize a fully dynamic and flexible composition [Benetallah 02].

We focus on two difficulties of current web service composition. One is that a workflow model is not good enough to represent composition of web service, where there include concurrently executable service components. In other words, describing compositions of web services as a workflow seems harder than describing them as a multi-agent scenario, where concurrently executable services can be easily represented. The other is that above frameworks do not have a function to insert semantics into the composite web service. The lack of this function not only restricts

J. Lee and M. Barley (Eds.): PRIMA 2003, LNAI 2891, pp. 98–109, 2003.

combinable services to several providers who have the same semantics, but also restricts the reusability of composite web services that was once made, to several requesters who understand its semantics. Although a large number of studies have been carried out on the latter problem, which is usually called an 'interoperability' problem [Martin 99][Nodine 98][Sycara 99], little attention has been given to the former one. We focus on these two problems.

In dealing with the former problem, we have developed a scenario description language *Q* and IPC (Interaction Pattern Card), which provide the environment where the web services can be easily composed in terms of a multi-agent scenario that controls agents' behaviors [Ishida 02b]. The advantage of applying this language to web service domain is twofold. One is that it enables users to easily describe web service composition, even if it includes concurrently executable services. The other is that it can handle multi-agent's scenario consisting of multiple service requesters who need cooperation as well as locally independent processing on the web.

Digital cities [Ishida 02a], which are being developed around the world, and collaborative commerce (C-commerce) [Behruens 00], which has recently become a hot issue in e-market place, are good examples of our objective domain where the scenario description works more effectively than the workflow model. Because there exist heterogeneous agents that behave concurrently as well as cooperatively in these domains, a multi-agent scenario will be woven through "interaction design" of which importance would be claimed in this paper.

As to the latter problem, we expect the Semantic Web [TBL 02] to give well-defined meaning to better enable computers and people to work in cooperation. This would be realized by marking up the contents of Web with well-defined semantics. Our work adopts DAML-S [Ankolokar et.al. 01] as service description language because it provides a semantically based view of a web services. Though this language enables agents to automatically discover, evaluate and invoke web services, it is too difficult for human users to write this language. Therefore, we introduce a translator, which translates the multi-agent web service scenario, written by our language, to the DAML-S, to put semantics on the scenario for the purpose of providing flexible reusability of once combined web services.

In this paper, we introduce a system which adopts a Semantic Web approach and scenario description language to guarantee interoperability and interaction design, respectively, and show their performance by applying them to the domain of composing web services. In the following sections, we discuss our philosophy of designing multi-agent system, which supports human user on the Web. Particularly, we stress the importance of "interaction design" among the agents. Section 3 introduces the scenario description language *Q*, and shows application domain where a multi-agent scenario would be necessary. Section 4 describes our proposed architecture for the web service composition and explains about the translator that adds semantics to the web service composition. We conclude in Section 5.

2 Design Philosophy

We consider three requirements that architecture of web service composition should fulfill. First, it should provide a framework for 'interaction design' among heterogeneous and legacy agents with different roles. Here, each agent plays a

different role but needs collaboration to achieve their respective goals. Second, it should guarantee 'interoperability' among heterogeneous agents from different platforms. Third, it should provide a framework to put semantics on the web service composition for the purpose of providing flexible reusability of once combined web services. Here, we will discuss previously developed systems by industries' initiative, and introduce our standpoint of designing web service architecture where "human is in the loop".

2.1 Current Web Services

Interaction Design in WSFL, XLANG, and BPEL4WS

Interaction design has been developed for web services, such as IBM Web Services Flow Language (WSFL), and Microsoft's XLANG. WSFL specifies interactions between web services which are modeled as links between endpoints of the web services' interfaces. Microsoft XLANG provides both the model of an orchestration of services as well as collaboration contracts between orchestrations. BPEL4WS [Virdell 03] provides a language for the formal specification of business processes and business interaction protocols. It is designed to ensure that differing business processes can understand one another in a web services environment, and that they can realize a dynamic composition, but its components are limited to a several static service providers.

In the framework mentioned above, interactions among the components of web services are described as a workflow. However, since concurrently executable service components are included within a composition, it is hard to represent their interactions as a workflow. In other words, it seems harder to describe compositions of web services as a workflow than as a multi-agent scenario described in an event driven manner where concurrently executable services can be easily represented through the 'interaction design' processes.

Interoperability in UDDI, WSDL, and SOAP

Web services are designed to provide interoperability between diverse applications. The platform and language independent interfaces of the web services allow the easy integration of heterogeneous systems. Web languages such as Universal Description, Discovery, and Integration (UDDI), Web Services Description Language (WSDL) and Simple Object Access Protocol (SOAP) define standards for service discovery, description and messaging protocols. In these frameworks, since the common interfaces are defined within the similar services, it is easy to find an alternative service to be combined. In addition, UDDI descriptions are augmented by a set of attribute, called tModels, which describe additional features such as the classification of services. However, these standards and protocols are available within several static service providers but don't contribute to an interoperability for a web service recomposition.

As mentioned above, current platform is hardly adequate neither for realizing complex composition of services on the Web, nor utilizing Web's resource to it's fullest. Notably, architecture for (1) unfolding E-commerce, which is carried out by multiple enterprises or binding dynamic collaboration processes to coordinate information and services of cities in an open environment, (2) registering composite

process, which is generated as a result of Web service composition, to the Semantic Web is needed.

2.2 Interoperability versus Interaction Design

The two approaches of interoperability and interaction design are complementary for multi-agent design (see Fig. 1).

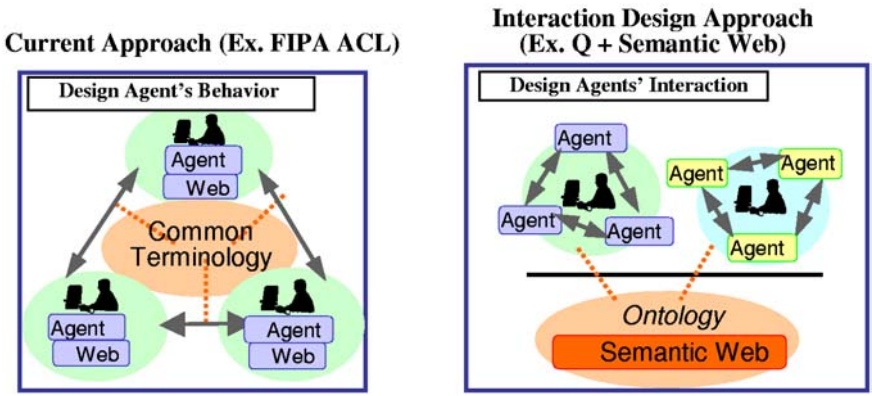


Fig. 1. Collaboration on the Web through Interaction and Interoperation

Interoperability

In order to guarantee interoperability among agents from different platforms, several specifications have been proposed by FIPA. Agents must be designed to be compliant with these specifications in order to interoperate with each other. Among these specifications, the Agent Communication Language (ACL) was designed to allow agents to communicate with each other. Therefore, we focus our discussion on ACL for interoperability among agents. Speech Act theory has been widely accepted in ACL as a way for designing agents that can react to a message, because an agents knows its illocutionary function while it may do not understand the propositional content. Doing so, ACL can be independent from content language. Therefore, a lot of ACL is based on speech act theory, including FIPA ACL and KQML [Finin 94].

FIPA ACL contains a set of message elements, including performative, sender, receiver, reply-to, content, language, encoding, ontology, protocol, conversation-id, reply-with, in-reply-to and reply-by. These elements realize interoperability among the very limited web services which are based on FIPA ACL framework.

Interaction Design

Interaction design has been widely studied for web services IBM WSFL specifies interactions between web services which are modeled as links between endpoints of the web services' interfaces. Microsoft XLANG provides both the model of an orchestration of services as well as collaboration contracts between orchestrations. BPEL4WS is designed to ensure that differing business processes can understand one another in a web services environment. Here, the interactions among the web service

components, which are regarded as the agents, are described as a worklow where it seems hard to describe the interaction among the concurrently executable agents.

We introduce *Q* to describe a *human request* to a large number of (*legacy*) *agents* with different roles [Ishida 02b]. Its primitives include *cue* and *action*. A sensing function is defined as a *cue*, and there is no side effect to sensing. An acting function is defined as an *action*, which may change the environment of agent system. The *cues* and *actions* are performed by agents, and their semantics is dependent on those agents. A Guarded command is for describing concurrent reactive behavior of agents. A scenario is described as state transitions. Each state is defined by a guarded command. The Scenario Description language can provide a clear interface between agent system developer, scenario writer and interaction designer.

Collaboration between Scenario Writer and Agent System Developer: The procedure of creating a new scenario is described as follows. First, the scenario writer and agent system developer agree upon the *cue*s and *action*s as an interface between them. Second, the scenario writer describes a scenario, while the agent system developer implements *cue*s and *action*s. The biggest effect of introducing *scenario* is that it can provide a clear interface between computer professionals and application designers, who have very different perspectives, and this effect would contribute to realize dynamic composition of web services without middle-agent frameworks. [Sycara 99] [Martin 99] [Nodine 98].

2.3 Complementary Approaches

As discussed above, the two approaches of interoperability and interaction design are complementary for multi-agent design, which means how to compose the web services.

Collaboration among Agents and Collaboration among Humans: The interoperability approach aims at collaboration among autonomous agents from different vendors and platforms, while the interaction design approach allows humans from different perspectives to collaborate when designing agents which crawl across the web on behalf of humans.

Interoperation among Agents and Interaction between Human Users and Agents: Agent communication language (ACL) aims at allowing agents to communicate with each other, while multi-agent scenarios describe the interaction between human users and agents. Agents constrained under scenarios can interoperate with each other by ACL.

Interoperation and Architecture Design of Agents: Composing web services can be considered as a multi-agent planning. Here, it is rather difficult for heterogeneous agents to decompose problems into sub-problems, allocate sub-problems, exchange sub-problem solutions and synthesize overall solutions autonoumously, even if the interoperation does work among the agents. Agents should be intelligent and autonomous in interoperability approach to achieve their goals. Whereas with the interaction design approach, multi-agent planning might be done by human users, who request the web services, by means of scenarios that coordinate agents activities.

In interaction design, agents can be autonomous or dependent. If autonomous, scenarios can be simple; if not, scenario writers should specify all details.

In the next section, we introduce the language we have been developed to describe multi-agent scenarios to realize web services composition.

3 Languages for Multi-agent Scenarios

3.1 Interaction Design Language: *Q*

The language *Q* is originally designed for describing scenarios of multi-agent's. For details see [Ishida 02b]. We currently re-designed *Q* for describing a variety of web service composition. In the multi-agent world, it requires to control multiple agents by the multi-agent scenario in the distributed environment. In the following section, we introduce the scenario description language *Q,* and discuss how multi-agent scenario is described for web service composition. The salient features of *Q*'s language functionality are summarized as follows. To explain the feature *Q*, we take an example of a simple Web service composition scenario as shown in Fig.2. Here, the Web service takes user's U.S. address to find the ZIP code and temperature.

Cues and Actions: An event that triggers interaction is called a *cue*. Cues are used to request agents to observe their environment. No cue is permitted to have any side effect. Cues keep on waiting for the event specified until the observation is completed successfully. Comparable to cues, *actions* are used to request agents to change their environment. The line starting with !verb denotes a request to the agent to make a certain action, while the line starting with ?verb denotes request to make a certain observation. Types of cues and actions are limited to ones given a priori, but unlike functions, the content of the cues and actions need not be defined since the interpreter carries out the interpretation and scenario execution. Thus, scenario can be said to have high readability, since it is complete in itself.

Scenarios: Guarded commands are introduced for the situation wherein we need to observe multiple cues simultaneously. A guarded command combines cues and actions. After one of the cues becomes true, the corresponding action is performed. A scenario is used for describing state transitions, where each state is defined as a guarded command. Scenarios can be called from other scenarios. Scenario describes state transitions using guarded command. Through scenarios, describing interactions according to states becomes possible. By using '*go sentences'* to define the next state, state transition can be described. If executing a group of actions end without specifying the next state, then it means the state transition is completed.

Agents: Each agent is defined by a scenario that specifies what the agent is to do. Even if a crowd of agents executes the same scenario, the agents exhibit different actions as they interact with their local environment (including other heterogeneous agents).

```
(defagent sample-agent
        sample-scenario)
   (defscenario sample-scenario (&pattern ($state "")
                                ($city "")
                                ($address "")
                                ($zipcode "")
                                ($temperature "")) 
    (scene1
     (#t
      (!open-input-window :label '("address" "city" "state"))
      (go scene2)))
    (scene2
     ((?push-button :button "OK")
      (!get-text :label "address" :result $address)
      (!get-text :label "city"    :result $city)
      (!get-text :label "state"   :result $state)
      (!webservice :wsdl-file
                   "http://webservices.eraserver.net/zipcoderesolver/zipcoderesolver.asmx?WSDL"
                :operation "ShortZipCode"
                :input (list $address $city $state)
                :extract '("ShortZipCodeResult")
                :result $zipcode)
      (!webservice :wsdl-file "http://www.ejse.com/WeatherService/Service.asmx?WSDL"
                :operation "getTemp"
                :input $zipcode
                :extract '("return")
                :result $temperature)
      (!display :data (list "Address:" $address $city $state
                        "Zip Code:" $zipcode
                        "Temperature:" $temperature))
      (go scene1))
     ((?push-button :label "End")
      (!display :data (list "Bye!")))))
```

Fig. 2. Multi-Agent Web Service Scenario

3.2 Application Domain

The language Q realizes a collaboration needed in C-commerce (Collaborative commerce), a next generation E-commerce (which is referred to as next generation E-commerce), and in Digital City.

Collaborative Commerce (C-Commerce) is the name given to commercial relationships carried out over a collaborative framework to integrate enterprises' business processes, share customer relationships and manage knowledge across enterprise boundaries [Behruens 00]. The ultimate aims of C-Commerce initiatives are to maximize return on intellectual capital investment, business agility and the quality of the customer experience. C-Commerce is far more crucial than basic B2B e-commerce that is designed to construct a virtual link for a pre-defined community of trading partners to buy or sell goods and services. While, the traditional web service frameworks cannot help C-Commerce focusing a one-dimensional transaction, there is no opportunity to negotiate enhancements to products in order to have them match unique customer needs and requirements. Lacking an open, reliable platform, traditional c-commerce vendors could not develop flexible, sharable drag-and-drop

modules among their products. Our proposed Semantic Web service architecture, which consists of two frameworks, such as "interaction design" and "interoperability", enables C-Commerce to offer new products, services and multi-dimensional collaboration which brings global enterprises a step closer to realizing the promise of increased velocity in supply chains and efficient inter-enterprise processes.

Digital Cities is to build an arena in which people in regional communities can interact and share knowledge, experience, and mutual interests [Ishida 02a]. Digital cities integrate urban information (both achievable and real time) and create public spaces in the Internet for people living/visiting the cities. Digital cities are being developed all over the world because the Web enables us to create rich information spaces for everyday life as well as the global businesses. While the Internet provides global information, life is inherently local and heterogeneous reflecting the different cultural background. Here, we don't need or have any standard of protocol for communication on the Web, but need some collaborative information sharing among different cities. Since this type of collaboration seems to be realized by local interaction among some cities which have a common background, rather than by global interoperation through the Web. For example, a project to link Digital City of Japan and China has commenced as part of Asia Broadband Plan. Coordination of the information retrieval agent, which is independently developed on Digital Cities, is a good example to show that application area of Web service composition is expanding.

Similar to C-Commerce, collaboration between Digital Cities requires a framework that can create cooperating task scenario, and execute that scenario using information provided on the Web.

4 Architecture for Multi-agent Semantic Web Services

In this section, we will explain how to convert scenario description to DAML-S, which put the semantics to existing web service composition scenario, enabling other users to flexibly recompose or reuse those services. In the following, we introduce our Semantic Web service architecture which consists of two main modules; one is for realizing Multi-agent scenario, and the other is for translating it to DAML-S description to add the semantics.

4.1 Architecture

By means of converting a web service scenario, which based on *Q,* to the representation of description logic, semantics of service composite process is described.

The description of composite process added semantics is called the 'process semantics'. The 'process semantics' enables other service requesters to utilize this service composition for a web service plan to be satisfy this requester's constraints, using concept layers provided by service ontology. We employed here DAML-S, as an ontology language that describes property and capability of web services. Figure 3 shows our web service architecture for creating, converting, and releasing web service scenario on the Web.

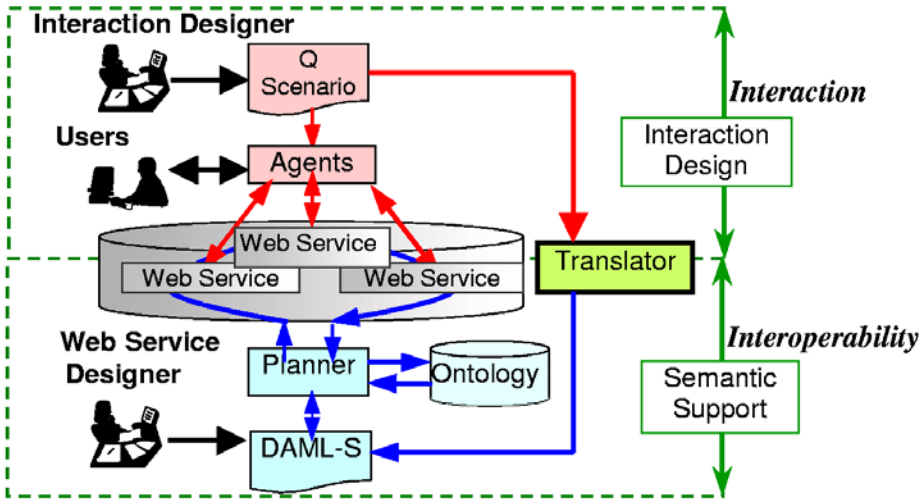


Fig. 3. Semantic Web Service Architecture for Composing Multi-agent Scenario

4.2 Scenario Execution by Multiple Agents

The executor for Web service scenario constitutes of Web service interface written in *Q* language interpreter and Scheme. Figure 4 shows the system architecture. Components surrounded by bold lines indicate Web service interface. Cues and actions described in the scenario is interpreted and executed by the interpreter. In case an action executing a Web service is given, Web service interface is called.

Information needed to execute Web service, such as URL in WSDL file, operation that is used, and value(s) to be sent to the service, are included in the interpreter's execution request. The interface first executes parser for analyzing WSDL. The parser obtains WSDL document from assigned URL, and then analyzes the document. When it finishes analyzing the document, outgoing SOAP message and address of SOAP server is extracted. Following this, the interface exchanges messages with Web service's SOAP server at SOAP transmitter/receptor. When the result is returned, the data is returned and then the procedure is terminated.

4.3 Scenario Translator to DAML-S Description

By describing service composition process in *Q* language, we have enabled users to easily construct a composite service. However, processing transaction between multiple, different Web services cannot be realized only with Web service composition scenarios. Moreover, since service scenarios lack reusability services accommodating situations of various users cannot be provided. By converting scenario descriptions, which reflect the needs of users in various situations, to DAML-S, reusing the scenario becomes possible through placing Web service composition mechanism on the server-side. In this section, the conversion of 'guarded command' to DAML-S process semantics is outlined. Guarded command is important

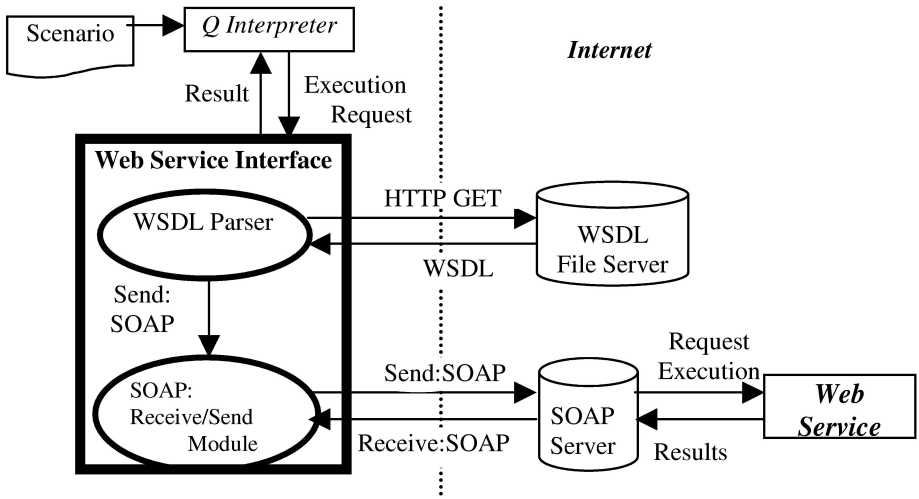


Fig. 4. Scenario Execution

in realizing multi-agent scenario, and converting multi-agent service scenario into DAML-S process semantics must be accurately performed. But we don't yet to examine the completeness of this translation.

Guarded command generally is expressed in multiple pairs of guard and statement. A guard consists of Boolean function and communication primitives such as transmission or reception. When Boolean function is true and communication primitive is ready to communicate, the guard is considered true. If guard is considered true inside the guarded command, then next execution line is executed. However, if multiple guards are true, only one guard is non-deterministically selected to execute the next execution line since multiple guards are evaluated in parallel.

In *Q* language, event-waiting '*cue*' corresponds to a guard, whereas outside '*action*' corresponds to execution line. As mentioned in 3.1, the event-waiting corresponding to guard is considered true when event is observed, and as a result, subsequent group of actions are executed.

Guarded Command

Rules to convert guarded commands, which consist of actions and event-waiting cues, to DAML-S process model will be discussed. Guarded commands are sequentially controlled and executed in the following steps.

1. Initiate parallel execution of multiple event-waiting classes (*cue*) using split class execution. (Event observation and execution processing unit)
2. Wait until at least one event among the observed events occurs using Repeat-Until class. (Event-waiting processing unit)
3. Select one event that has occurred, and execute Action Set class that is triggered by that event, using Choice class. (Event selection and execution processing unit)

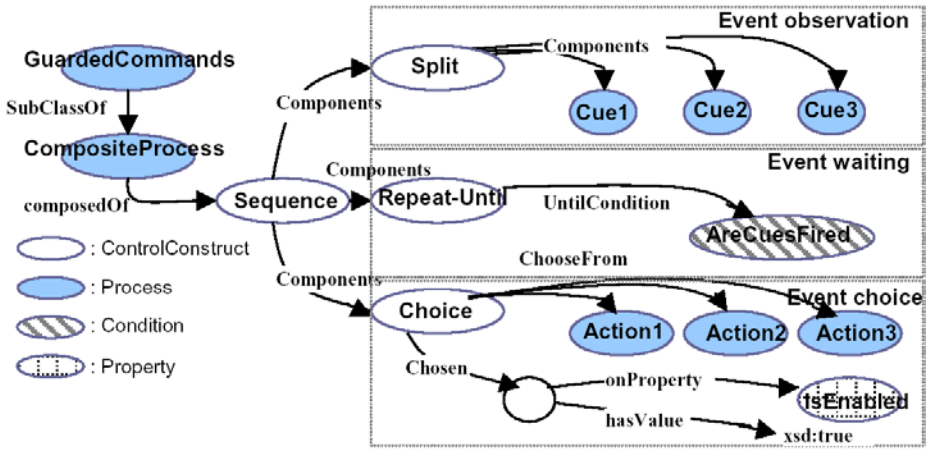


Fig. 5. RDF directed graph of Guarded command based on DAML-S Control Structure

Using control structure class called Sequence class provided in DAML-S, guarded commands are converted to subclass of Composite Process class, which sequentially executes event observation and execution processing unit, event-waiting processing unit, and event selection and execution processing unit. Figure 5 shows DAML-S expression of guarded commands, which consist of event observation and execution processing unit, event-waiting processing unit, and event selection and execution processing unit, in RDF's directed graph.

5 Conclusion

Our research aims at realizing social knowledge through sharing and employing information on the Semantic Web. As a first step, we proposed a scenario description language *Q,* which can describe multi-agents' Web service composition process through multi-agent scenarios. This language provides the framework for 'interaction design', which enables us to design interaction between users, between users and agents, and between agents, as well. Furthermore, we provided a framework that enables other users to flexibly reuse composite services by developing a translator that converts multi-agent scenario into DAML-S and a feature that registers services to the Semantic Web as a new service. By employing this translator, 'interoperability' is guaranteed on the composite web service, and consequently it becomes reusable to other users. The concepts of 'interaction design' and ' interoperability' work complementarily for coordinating heterogeneous agents and humans who have different motivations. This overall architecture not only enhances collaboration between users on the Web, but also facilitates user-agent interaction design, and therefore is expected to contribute to Collaborative Commerce and Digital City that will unfold in the Internet in the future.

References

[Ankolokar et.al. 01] A. Ankolokar et.al., DAML-S: Semantic markup for web services. *In Int. Semantic Web Working Symposium*, pages 411–430, 2001

[Arkin et.al. 02] Assaf Arkin et. Al, Web Service Choreography Interface (WSCI) 1.0, http://www.w3.org/TR/wsci/, 2002.

[Behruens 00] Stefan Behruens, Electronic Collaborative Commerce – An Overview of Enabling Technologies – Seminar Paper, Schloss Reichartshausen am Rhein, European Business School, 2000.

[Benetallah 02] B. Benatallah, M. Dumas, M.-C. Fauvet, and F. Rabhi. Towards Patterns of Web Services Composition. In S. Gorlatch and F. Rabhi, editors, Patterns and Skeletons for Parallel and Distributed Computing. Springer Verlag, UK, Nov 2002.

[Box 00] D. Box, D. Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn, H.F. Nielsen, S. Thatte, and D.Winer. Simple Object Access Protocol (SOAP) 1.1. http://www.w3.org/TR/2000/NOTE-SOAP-20000508/, 2000.

[Christensen 01] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana. Web Services Desctiption Language (WSDL) 1.1. http://www.w3.org/TR/wsdl, 2001.

[Finin 94] Finin, T., Fritzson, R., McKay, D., McEntire, R.: KQML as an Agent Communication Lnguage. *International Conference on Information and Knowledge Management (CIKM-94)*, 1994

[Ishida 02a] Ishida, T.: Digital City Kyoto: Social Information Infrastructure for Everyday Life. *Communications of the ACM (CACM)*, Vol. 45, No. 7, pp. 76–81, 2002

[Ishida 02b] Ishida, T.: *Q:* A Scenario Description Language for Interactive Agents. *IEEE Computer*, Vol.35, No. 11, pp. 54–59, 2002

[Martin 99] D. Martin, A.Cheyer, and D. Moran, The Open Agent Architecture; A framework for building distributed software systems. Applied Artificial Intelligence, 13(1-2): pages 92–128, 1999.

[Nodine 98] M. Nodine and A. Unruh, Facilitating Open Communication in Agent Systems. In M. Singh, A. Rao, and M. Wooldridge, editors, Intelligent Agent IV: Agent Theories, Architectures, and Languages, pages 281–296, Springer-Verlag, 1998.

[Sycara 99] K. Sycara, M.Klusch, S. Widoff, and J. Lu, Dynamic service matchmaking among agents in open information environments, *SIGMOD Record (ACM Special Interests Group on Management of Data),* 28(1): pages 47–53, 1999.

[TBL 02] T. Berners-Lee, J. Hendler, and O. Lassila, The Semantic Web. Scientific American, 284(5): pages 34–43, 2001.

[UDDI 02] The Evolution of UDDI Technical White Paper, http://www.uddi.org/pubs/, 2002.

[Virdell 03] Margie Virdell, Business Processes and Workflow in the Web services world, - A workflow is only as good as the business process underneath it www-106.ibm.com/developerworks/webservices/library/ws-work.html

A Generic Model for Distributed Real-Time Scheduling Based on Dynamic Heterogeneous Data

Peter Bloodsworth, Sue Greenwood, and John Nealon

Department of Computing,
Oxford Brookes University, Oxford, UK,
{pbloodsworth,sgreenwood,johnnealon}@brookes.ac.uk
http://cms.brookes.ac.uk/computing

Abstract. In recent years multi-agent systems have demonstrated the power to handle complex real-time scheduling problems. They have also been used to integrate distributed information to provide access to heterogeneous data sources. Combining these two approaches to produce a multi-agent system that can use distributed heterogeneous data for real-time scheduling is a challenging problem. In this paper we introduce the concept of a generic model, which can then be used to develop multi-agent systems with these capabilities. We suggest that a model of this type can lead to improvements in scheduling performance and make multi-agent systems applicable to a range of new problem domains.

1 Introduction

The ability of multi-agent systems to handle complex real-world problems in a distributed manner has led to many important and diverse applications in recent years [1]. Research has demonstrated that multi-agent systems are well suited to monitoring incoming data and acting in an autonomous manner to schedule actions in order to solve complex real-world problems [2]. This being the case it is no surprise that agents are being applied today in a number of problem domains where real-time scheduling is desirable. These domains include manufacturing [3], air traffic control [1,4], transportation systems [5,6] and organizational scheduling [7] as well as many others. In order to develop multi-agent systems in these and other areas access to real-world information is often necessary.

The Internet is a primary source of heterogeneous data, it provides vast, ever expanding and dynamic real-world data organized in a highly distributed way [8]. It is all very well having such sources of data, but the problem that persists is how this data can be accessed and used. At the current time the Internet is structured in a manner that is biased against computer systems [9]. Web-content is still predominantly written using mark-up languages that concentrate on page appearance and formatting with no semantic information being made available. It is hoped that gradually XML will address this but we are not there yet. Giving multi-agent systems access to dynamic heterogeneous data is clearly a challenging problem for all of the reasons above. The potential for an increased role of

J. Lee and M. Barley (Eds.): PRIMA 2003, LNAI 2891, pp. 110–121, 2003.

multi-agent systems in problem domains, where the main focus is scheduling, provides a very persuasive reason for our work. Not only can improved scheduling performance result from better quality information but it is also hoped that new application areas may be opened up [10]. In order to accomplish these goals we suggest the development of a generic model that structures the capturing and use of heterogeneous data for real-time scheduling within multi-agent systems. In the next section we briefly consider the current state of research in this area.

2 Background and Previous Research

In recent years several multi-agent systems have been developed demonstrating the ability to access heterogeneous data as information agents. Other systems have been developed that can schedule in real-time. These two classes of multi-agent systems include the Oasis air traffic control project [11], the Warren portfolio management system [12], TravelPlan [6] and InfoSleuth [13] as well as many others [14,15,16,17,18,19]. Taking Oasis and Warren as examples that are indicative of current research we will now consider each in turn and identify their strengths and weaknesses. In each case we will demonstrate how each system could be enhanced by integrated approaches to planning and data handling.

Oasis is designed to aid decision-making by optimally scheduling aircraft landing slots in a busy airport. These actions involve the use of advanced real-time scheduling algorithms in addition to robust constraint management. Oasis provides an example of a multi-agent system that uses highly structured and reliable data in order to carry out its scheduling tasks in real-time. It receives aircraft location data and wind speed information from local sensors and then uses this to produce a schedule for landing slots. The scheduling of landing and takeoff slots are mainly dependent on when aircraft are ready, the weather conditions and the status of the runway. In order to obtain the maximum capacity from the airport and avoid delays the separation time between landing and take off slots must be minimized within certain safety bounds. If Oasis could access information from other airport systems then it could include these into the scheduling process thus improving the schedules produced. However doing this would require the integration of heterogeneous data. We will now consider a system, which demonstrates a method of achieving this.

Warren captures and then uses dynamic heterogeneous data in order to manage stock portfolios. It uses the Internet as its primary source of information. Stock prices and business news regarding companies in the users portfolio are monitored for changes and indications regarding the future performance of the stocks. The news regarding a company is assessed, aggregated and then fed back into the decision making process. Finally the system produces recommendations that are passed on to the user for a final decision. The major limitation with Warren is that financial markets change in real-time. In order for the system to deliver up to date information in a timely fashion it is important to be aware of real-time constraints [20,21]. This is especially true here because information in the financial domain is often only of value for a short period of time. Essentially

the system is addressing a scheduling problem, it is scheduling the correct time to buy and sell shares. If Warren could be adapted so that it could schedule in real-time it might be possible for it to trade on the user's behalf. At the very least the quality of information provided to the user would be drastically improved because stock tracking would happen in real-time.

Our analysis of Oasis and Warren might lead to the conclusion that each system would benefit from the abilities of the other. This situation highlights a problem that currently has yet to be addressed within multi-agent research. How do we integrate the abilities of distributed scheduling in real-time with access to distributed heterogeneous data sources? Accomplishing this might at first seem trivial, simply merging the two systems to form a hybrid with the attributes of both. On closer inspection it becomes evident that there is more to a solution than this. We will see that several new problems appear when designing a unified model. The following section will discuss these challenges in more detail.

3 Challenges in Developing the Model

The first challenge that we face is collecting the required information from heterogeneous systems. Our model will require the ability to draw information from a wide range of systems including the Internet, external databases and real-time information feeds. The Internet poses the greatest challenge in this regard because the data is presented in a semi-structured manner. Semi-structured data components are difficult to relate to one another making it very easy to collect the wrong information. XML promises to address many of these problems by separating the formatting and appearance information from the data. However it may be some time before XML becomes widely used and even then it is unlikely that all Internet content providers will adopt it [22]. With these considerations in mind we believe that the model should have methods for handling both XML and other formats such as HTML.

There have been several attempts at overcoming the problems associated with HTML many involving the construction of wrappers for data sources [23, 24,25,26,27]. Wrappers have generally been hand coded to provide access to unstructured data on the Internet. The dynamic nature of the Internet however can cause wrapper-based systems to fail [23]. This often happens when a website changes just enough to make the wrapper malfunction. In such cases the wrapper will have to be re-written in order to work with the new site format. It is not always possible to do this when a system is accessing a large number of web resources, we could end up constantly re-writing wrappers as websites change. It is also clear that this is not a practical solution in a real-time system where the wait for a wrapper to be re-written would render the system useless. Once the data has been collected we then need to be able to reason on it.

Humans are good at combining information from a variety of sources in order to solve scheduling problems. For example, consider the following situation. The bus you need to catch is overdue and you have a meeting at 4:30pm, do you walk or wait for the bus? From experience you know that traffic caused by parents

collecting children from school often causes the bus to be late at this time. To avoid missing the meeting you decide to walk. You get halfway there when it starts raining, remembering that the forecast was for heavy showers you decide to walk to the next bus stop and catch the bus, the meeting is not very important and not worth getting wet for! This is a simple example of how a person alters real-time schedules in response to a whole host of data. Here the human subject is integrating bus timetables, weather forecasts, geographical knowledge, the urgency of a meeting and local experience together to form a plan. There is also an amount of deduction going on, for instance combining the facts that the bus is late with the knowledge regarding traffic to conclude that you will miss your meeting if you wait for the bus. Drawing new conclusions and data from old to produce a form of "rich data" is a difficult task for a computer [28,29]. It is vital however, that some measure of information integration goes on, in order to provide the best quality information to base a schedule on.

Distributed real-time scheduling using a wide range of data is far from easy. There are several potential problems, for example how do we select the data to use for a particular scheduling task? Should all information sources have equal weight, or should some have more importance than others? Another important consideration is that real-time scheduling requires good quality real-time information, that is data with as few errors and inconsistencies as possible. It is likely that the information coming into the system at some stage will be fuzzy, incomplete or even conflicting. Any one of these difficulties can cause deadlocks to occur as one scheduling process has to wait for another or a stalled process. This has the potential to destroy the responsiveness of the system. Recent work [30] however has shown that these problems can be overcome by using a hybrid model of reasoning in uncertain domains. Another method of increasing the responsiveness of the system is to interleave pre-planned and real-time schedules [31].

It will be difficult to provide the system with methods for interleaving real-time and pre-planned schedules, however we believe that this is necessary in order to respond to events in a timely fashion. Producing optimal schedules tends to be very computationally demanding and thus will often take much more time. Pre-planned schedules are ready to be used at a moment's notice, but in order to do this they use cached information which may be less accurate. It is clear that there is a need to maintain the correct balance between the accuracy of optimal schedules and the speed of sub-optimal ones. By doing this we aim to produce schedules that are "good enough" within the demands of the individual problem domain [32].

The model also requires a method of prioritizing messages for dispatch to the user, this is necessary to avoid important messages being delayed by less important ones. Determining which message should be sent first is challenging but vital as the following example shows. Consider which message you would want first; the light is on in the bathroom or the house is on fire! It is clear that in real-time domains messages and actions must be prioritized in such a way that important messages and actions do not get held up by a queue of less important

ones. It is also important that a failure in sending a message or carrying out an action does not result in the deadlock we considered in a previous paragraph. A means of determining a hierarchy of priority is necessary to make this possible. Such a hierarchy must be as flexible as it is reasonable to produce, because the importance of messages often changes over time in dynamic environments.

The final notable challenge is the development of an agent architecture that is compatible with the model. The structure of the multi-agent system and hence the agent architecture must reflect the real-time nature of the problem domain. It will be necessary for the system to avoid bottlenecks in the communication process and work in such a way that the value of real-time information is not lost. There are sections of a system of this nature that will need to be able to work very quickly in order for the systems performance to be acceptable. Stalled processes must be avoided at all costs and there will need to be strong error checking and management capabilities.

4 Why Is a Generic Model Required?

Current research has provided many different system designs and architectures that tend to be heavily domain specific. Work in this area has so far provided a range of new system models for each individual application domain [1]. This has mainly been because so many of the methods employed were experimental. As we reach a greater understanding of the issues involved in developing systems of these types, it would be helpful if there existed a structure that could be reused to generate new applications of the technology. The aim is to build a generic model with the source of the distributed information not simply being limited to the Internet. The proposed model will allow multi-agent systems to monitor real-world events and changes through distributed sources of information and act in a flexible dynamic way to schedule and re-schedule actions in real-time.

The model aims to aid future research by providing a clear structure of the elements required in systems of this type and creating a common basis for further investigation. A general standardized model will provide an outline of what is required when constructing a multi-agent system of this nature. This could in time lead to a number of system deployments in a range of problem domains. A standardized approach can encourage greater collaboration between systems in different domains thus allowing a system to be used by other agent-based systems. The Agentcities project [33] has developed as an example of a possible deployment platform in which multi-agent systems work together. A standardized model also makes it possible to construct toolkits that ease system development. We will now consider the model in greater detail.

5 An Outline of the Proposed Model

In this section we present an initial high-level discussion of the model. This should be seen as work in progress, so far we have identified many of the problems that exist and have developed an initial model structure that is designed to

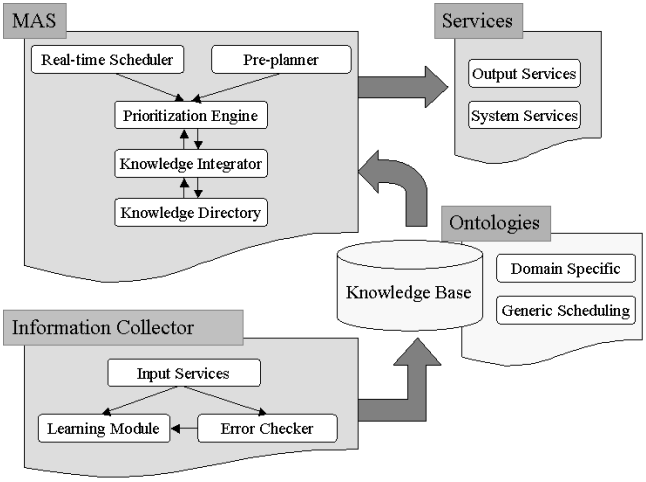


Fig. 1. The Proposed Model

address them. Here we suggest the role of each element of the model and briefly comment on how each should be implemented. It has also been decided which functions should be part of the multi-agent system and what should remain separate. These decisions have been based on achieving a balance between the functions that the multi-agent system will handle and the external services that should remain independent for reasons of stability and performance.

5.1 The Information Collector

It is clear that the entire system will rely on the information collector to provide good quality data within the available time constraints. Should it fail then the entire system is also likely to be seriously affected. One possible cause of a failure would be if the required data the system is looking for were moved to a new location within a domain. It will therefore be necessary for the data collector to search for a new URL should it change for some reason. We believe that a spider-like system can be created to accomplish this. The more robust the information collector is the better it is for system stability. Therefore it would be good to have a collection method with learning abilities. This could be achieved by using learning techniques that adapt to changes in the information source. In order for the learning algorithms to function correctly it will be necessary to have a strong error checking subsystem.

The error checker is required in order to ensure system stability [34]. It also allows the learning methods of the data collector to function correctly. As the learning modules try different methods of collecting data the error checker is vitally important in determining whether it has worked or not. This subsystem will make use of heuristics and statistical methods in order determine the correctness of collected data. The central ontology as used by the input services

functions can also be harnessed to determine possible errors by looking at the structure of the data as a whole. Where additional information is available it can be used to test the collected data to see if it is correct. For instance if you are collecting real-time train running information from the Internet it can be compared with the main train timetables to check that they are consistent. If the two sources disagree then you may conclude that there is an error in the data collecting process.

The input services part of the model is responsible for collecting information from many sources including the Internet and databases. As we discussed earlier this is a difficult problem especially on the Internet. Identifying the relevant information along with collecting and organizing data possibly present the greatest challenges. It is also vital we make sure related data is grouped and that cohesion is not lost. Wrappers have been used to accomplish this in the past but as we demonstrated earlier these have difficulty coping with the dynamic nature of websites and their structure [23]. Our proposed design places an ontology at the centre of this process [35]. It is then used to describe and thus recognize the relevant data whilst retaining the continuity and structure of it. The research will also investigate the use of heuristic or statistical methods to strengthen and enhance this process.

5.2 The Knowledge Base and Ontologies

The knowledge base represents all the knowledge that is available to the system. This includes databases that have been gathered by the information collector, in addition to this there could also be links to other databases and live source feeds from external sensors. The system services provide methods by which access to external databases and live data feeds can be made. As well as interfaces these also provide other basic system functions such as file access and SQL formatting. The knowledge integrator that we will consider in the next section, is responsible for merging all of these sources of data together to produce more detailed information.

The model will rely heavily on the use of ontologies to structure data access. They will provide a clear basis for agent communication and environmental representation. We envisage that both a generic scheduling and a domain specific ontology are required. The generic scheduling ontology will define the scheduling process that is common to all problem domains. The domain specific ontology will aid knowledge integration by relating the different types of available data to one another thus forming a more detailed picture of the current environment. Another use of this can be applied to the processes of information gathering and error checking.

5.3 The Multi-agent System

The knowledge integration section of the multi-agent system is responsible for drawing the complete range of available information together in order to make scheduling decisions. It is important that as much information as possible is

drawn from the collected data. As humans we often merge information from many sources in order to solve complex problems. The decision making process is frequently enhanced by the use of extra facts or beliefs that are derived from combining existing knowledge. For example if we know that the train is late departing from Cardiff and we also know that it stops at Bristol Parkway and finally London Paddington, then we can conclude that it will probably be late at Bristol. If we were catching the train at Bristol these thought processes might cause us to change our travel plans, we could perhaps choose an alternative route, which is quicker. This demonstrates the power of "rich" data within the scheduling process. Knowledge integration is necessary to provide the multi-agent system with the most complete picture of its environment. Our approach is to make the ontology central to this process, using it to define the relationship between different data sets. It is proposed that improved environmental awareness will lead to a better scheduling performance.

The real-time scheduler is one of the important aspects of the model. Here we have decided not to attempt optimal scheduling at this time, instead we aim to produce acceptable schedules within the constraints of the problem domain. This decision was taken because of the computationally demanding nature of generating optimal solutions and the fear that this could degrade the real-time abilities of the model. Recent work by Graham [36] has demonstrated that theory designed to schedule tasks in operating systems can be adapted to work within multi-agent systems. We aim to integrate this and other approaches to provide a solution in this area. It will be necessary to include time constraint management within the scheduling process. This will mean that if a schedule cannot be created within a very tight time frame then the system can switch to an instantly available pre-planned schedule. By interleaving real-time and pre-planned schedulers we aim to provide a truly responsive system.

A prioritization engine is required to ensure the balance of the system. This will prioritize outgoing messages to the user so that more trivial ones do not delay important ones. The engine will be used to interleave pre-planned and real-time schedules assigning priority to the correct method within the time constraints. Another area where the engine may be used is in determining a hierarchy of events coming into the system. The system needs to have a method of sorting important events from more trivial ones. This will ensure that more important information is acted on quickly thus making the most of the real-time scheduling. All of these features will require the development of a method of determining the priority of events or actions. The central ontology plays another part here by defining the conditions for each level of priority.

The pre-planner section of the model sits in the background producing plans in advance for a range of possible occurrences. It will pre-plan for the most common and most difficult eventualities. By having a set of plans ready to use at a moments notice the system can continue to perform under the most difficult of time constraints. It is obvious that pre-planned schedules will not be as accurate. Such plans will be based on less up-to-date information and could lead to poorer performance. This being the case pre-planned schedules must only be used in

situations where the real-time scheduler cannot generate in time. Judging these decisions will be a very difficult task, it is clear that it may not be possible to be right on every occasion. When designing this section of the model a measure of caution is necessary in order to avoid making decisions that are too close to the time limit. We do not want situations occurring where the system waits so long for a real-time schedule that the slightest delay in delivering it to the user will invalidate it.

The primary objective of the knowledge directory is to keep track of all the available sources of information. In addition to this it will also provide details of the access methods that are required for each information source. This aspect is vital because there are so many knowledge sources that require different access methods, for example XML, HTML or SQL. The knowledge directory will be used by agents to look for sources of information and identify correct access methods for these. The directory will also log the data that is currently available, by doing this we aim to prevent a situation where the system stalls while it waits for a necessary piece of information to become available. It is clear that when data is being used from a range of Internet and other sources there is always a certain level of fluidity, as sources become unavailable due to external factors. We expect the knowledge directory to provide a robust level of flexibility that reduces the systems exposure to common errors.

5.4 Services

Finally we turn to output services, we can think of these as representing anything that interacts with the end-user to supply schedules. These will include web-based interfaces, WAP, SMS, email or other methods of communication. It is important that all of the available methods are reliable, because we do not want delays in message sending to impair the real-time abilities of the system. It will require the use of robust error checking in order to prevent communication failure stalling the system.

6 Future Work

The first stage, having completed the work on the information collector, is to finish designing a multi-agent architecture that is compatible with the model. At this point it will be necessary to develop a prototype implementing the multi-agent architecture. There are many candidate problem domains in which this prototype can be developed, one of constrained complexity will be chosen. By doing this we aim to test and refine the model and the architecture. This may lead to further improvements in its performance as we strengthen some of the heuristics governing the decision making process. Having completed this section we will move on to developing a system for a more complex problem domain. The area of transportation planning and management may be used as an application for a more advanced version of our system.

Travel has been a constant source of frustration and anxiety for many people in Britain over the last few years. Nowhere has this been more so than on the railways where there has been short notice delays and cancellations often leaving travellers confused and wondering what to do next. In recent months rail companies have started to post real-time train running information on their websites in a variety of formats. Unfortunately few of us yet have access to the Internet when we are standing on a platform. All these properties make travel a good problem for thoroughly testing systems developed from our model. The system will monitor the latest real-time railway information and then act on behalf of its users to reschedule their journey when it is necessary.

Users would initially register via a website or a mobile phone (using WAP or SMS) their intention to travel. The multi-agent system would then monitor their journey at each stage notifying them if a train is late or cancelled using SMS text messaging. The system will collect heterogeneous real-time train running information from the different train company websites and integrate this with timetable and other knowledge sources in order to monitor the journey. In the event that a train is cancelled or heavily delayed the system would analyse alternative routes and inform the user of the most efficient option using SMS/WAP. Obviously when the system plans alternatives it will use its integrated knowledge of real-time train running information to avoid other problem routes. Following the testing and evaluation of this application we will confirm that the model is generic in nature. We shall do this by implementing it in a wide range of problem domains.

7 Conclusion

In this paper we have shown that a generic model for distributed real-time scheduling based on dynamic heterogeneous data will open the way for a range of powerful new systems. We have suggested the elements that a model of this nature requires and how they can be organized structurally. It is clear that there are many difficulties to be addressed, these include the collecting of data from heterogeneous sources, information integration, real-time scheduling and the prioritization of actions. Through our research we have seen that placing ontologies at the centre of the model can help us to overcome many of these challenges. Our research is ongoing with a prototype system derived from the research near to completion. Further work will be presented in due course as our work proceeds.

References

1. Nicholas R. Jennings, Katia Sycara, and Michael Wooldridge, A roadmap of agent research and development, Autonomous Agents and Multi-Agent Systems, Volume 1, pages 7–38, 1998.
2. Travis Bauer and David Leake, A Research Agent Architecture for Real Time Data Collection and Analysis, Proceedings of the Workshop on Infrastructure for Agents, MAS and Scalable MAS, 2001.

3. K. Fischer, An Agent-based approach to holonic manufacturing systems, In Information technology or balanced automation systems in manufacturing (Kluwer Academic Publisher), pages 3–12, 1998.
4. M. Georgeff and A. Rao, BDI Agents from Theory to Practice, In Proceedings of ICMAS , pages 312–319, 1995.
5. K. Fischer, J.P. Miller and M. Pischel, Cooperative Transportation Scheduling: an Application Domain for DAI, In Journal of Applied Artificial Intelligence, Vol 10; Number 1, pages 1–34, 1996.
6. D. Borrajo, D. Camacho and J. M. Molina, Travelplan: A multiagent system to solve web electronic travel problems, In ACM Workshop on Agent-Based Recommender Systems, Fourth International Conference on Autonomous Agents, Barcelona, 2000.
7. Hans Chalupsky et al, Electric elves: Applying agent technology to support human organizations, In Proceedings of IAAI-2001, Seattle, 2001.
8. T.R. Payne, R. Singh, and K. Sycara, RCAL: A Case Study on Semantic Web Agents, In Proceedings of Autonomous Agents and Multiagent Systems (AAMAS 02), ACM Press, NewYork, 2002.
9. R. Doorenbos, N. Kushmerick and D. Weld, Wrapper induction for information extraction, In Proceedings of the International Joint Conference On Artificial Intelligence, Vol 15; Number 1, pages 729–737, 1997.
10. Laurent Magnin, Jian-Yun Nie and Hicham Snoussi, Heterogeneous Web Data Extraction using Ontology, University of Montreal, 2001.
11. Magnus Ljungberg and Andrew Lucas, The oasis air-traffic management system, In Proceedings of the Second Pacific Rim International Conference on Artificial Intelligence, PRICAI '92, Seoul, 1992.
12. K. Decker, K. Sycara, and D. Zeng, Designing a multi-agent portfolio management system, In Proceedings of the AAAI Workshop on Internet Information Systems, pages 26–34, 1996.
13. R.J. Bayardo et.al, InfoSleuth: Agent-Based Semantic Integration of Information in Open and Dynamic Environments, In Proceedings of ACM SIGMOD, Vol 26; Number 2, pages 195–206, 1997.
14. B. Das and D. Kocur, Experiments in Using Agent-Based Retrieval from Distributed and Heterogeneous Databases, IEEE Knowledge And Data Engineering Exchange Workshop, pages 27–35, 1997.
15. A. Mian, N.G. Shaw and S.B. Yadav, A comprehensive agent-based architecture for intelligent information retrieval in a distributed heterogeneous environment, Decision Support Systems Vol 32, Number 4, pages 401–415, 2002.
16. S.T. Yuan, A personalized and integrative comparison-shopping engine and its applications, In Decision Support Systems, Vol 34, Number 2, pages: 139–156, 2003.
17. T. Ito, N. Fukuta, T. Shintani, and K. Sycara, BiddingBot: A multiagent support system for cooperative bidding in multiple auctions, In Proceedings of the Fourth International Conference on Multi-Agent Systems (ICMAS'2000), Boston, USA, pages 399–400, 2000.
18. K.L. Clark and V.S. Lazarou, A Multi-Agent System for Distributed Information Retrieval on the World Wide Web, In Proceedings Of The Workshops On Enabling Technologies, Vol 6, pages 87–93, 1997.
19. Ricardo Aler, Daniel Borrajo, David Camacho and José Manuel Molina, Solving travel problems by integrating web information with planning, In Foundations of Intelligent Systems, Number LNAI 2366 in Lecture Notes in Artificial Intelligence, pages 482–490, 2002.

20. D.J. Musliner et al, The Challenges of Real-Time AI, IEEE Computer Society, Vol 28; Number 1, page 58, 1995.
21. Bryan Horling, Victor Lesser, Regis Vincent, and Tom Wagner, The soft real-time agent control architecture, Technical Report – American Association For Artificial Intelligence Ws, pages 54–65, 2002.
22. José Luis Arjona, Rafael Corchuelo and Miguel Toro, Automatic Extraction of Information from the Web, 2002.
23. S. Minton, I. Muslea and C. Knoblock, A hierarchical approach to wrapper induction, In Proceedings of the Third International Conference on Autonomous Agents (Agents'99), pages 190–197, 1999.
24. Jean-Robert Gruser et al, Wrapper Generation for Web Accessible Data Sources, Proceedings Of The IFCIS International Conference On Cooperative Information Systems, pages 14–23, 1998.
25. B. Rahardjo and R. Yap, Automatic Information Extraction from Web Pages, Research and development in information retrieval; Proceedings of the 24th international ACM SIGIR conference, Vol 24, pages 430–431.
26. Sudarshan Chawathe, Hector Garcia-Molina, Joachim Hammer, Kelly Ireland, Yannis Papakonstantinou, Jeffrey D. Ullman and Jennifer Widom, The TSIMMIS Project: Integration of heterogeneous information sources, In the 16th Meeting of the Information Processing Society of Japan, Tokyo, pages 7–18, 1994.
27. Laura Bright, Jean-Robert Gruser, Louiqa Raschid and M. E. Vidal, Wrapper Generation for Web Accessible Data Sources, In Proceedings of Conference on Cooperative Information Systems, pages 14–23, 1998.
28. T.G. Harvey, Problems in Heterogeneous Knowledge Source Integration: TraumaGEN and the Metathesaurus, Technical Report – American Association For Artificial Intelligence Ws, Volume 98-04, pages 28–34, 1998.
29. M.N. Huhns and M.P. Singh, Multiagent systems in information-rich environments, In Cooperative Information Agents II, Volume 1435 of LNAI, pages 79–93, 1998.
30. X. Luo, C. Zhang, and N.R. Jennings, A hybrid model for sharing information between fuzzy uncertain and default reasoning models in multi-agent systems, Int Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 10 (4), pages 401–450, 2002.
31. K. Decker and T. Harvey , Planning Ahead to Provide Scheduler Choice, University of Delaware, 2001.
32. A. Garvey and V. Lesser, Design-to-time Real-Time Scheduling, In IEEE Transactions of Systems, Man, and Cybernetics, Vol 23, No 6, 1993.
33. S.N. Willmott et al, Agentcities: A Worldwide Open Agent Network, Agentlink News Number 8, pages 13–15, 2001.
34. Thomas Eiter, Viviana Mascardi and V.S. Subrahmanian, Error-Tolerant Agents, Lecture Notes In Computer Science, Computational Logic: Logic Programming and Beyond, pages 586–625, 2002.
35. Vasant Honavar, Jaime Reinoso-Castillo and Adrian Silvescu, Ontology-Driven Information Extraction and Knowledge Acquisition from Heterogeneous, Distributed, Autonomous Biological Data Sources, Iowa State University, 2001.
36. John R. Graham, Real-time Scheduling in Distributed Multiagent Systems, PhD Thesis, University of Delaware, 2001.

SweMas: Toward a Practical Multi-agent Framework Utilizing the Semantic Web

Jaeho Lee

Dept. of Electrical and Computer Engineering, The University of Seoul
90 Cheonnong-dong, Tongdaemun-gu, Seoul 130-743, Korea
jaeho@uos.ac.kr

Abstract. The Semantic Web aims to represent the data on the World Wide Web to have formal semantics that will enable autonomous agents to reason about the data and carry out more intelligent tasks on behalf of the user. Especially the OWL Web Ontology Language is designed for use by applications that need to process the content of information instead of just presenting information to humans. OWL facilitates greater machine readability of Web contents than that supported by XML, RDF, and RDF Schema by providing additional vocabulary along with a formal semantics. Autonomous agents utilize the formal semantics to autonomously interact with other agents. In this paper we present our approach toward a multi-agent framework to utilize the Semantic Web. The framework called SweMas has interfaces to the Semantic Web ontology service and a specialized inference service. The framework also has a distinguished component to transform between ontologies. These interfaces to the Semantic Web services are built on the FIPA-compliant JADE agent framework. In our framework JADE serves as a middleware to support agent management, agent communication, and agent interaction protocols. JADE also provides an environment for application agents to be developed and get plugged in to the framework. We lay out the architecture of our multi-agent framework to utilize the Semantic Web and the rationales for our design of the framework.

1 Introduction

The Semantic Web aims to represent the data on the World Wide Web to have formal semantics that will enable autonomous agents to reason about the data and carry out more intelligent tasks on behalf of the user [9,2,28]. Especially the OWL Web Ontology Language is designed for use by applications that need to process the content of information instead of just presenting information to humans. OWL facilitates greater machine readability of Web contents than that supported by XML, RDF, and RDF Schema by providing additional vocabulary along with a formal semantics [29]. Autonomous agents utilize the formal semantics to autonomously interact with other agents. In this paper we present our approach toward a multi-agent framework to utilize the Semantic Web.

As shown in Figure 1, the framework called SweMas[1] has interfaces to the Semantic Web ontology service and a specialized inference service. The framework also has a

[1] SweMas is an acronym for **S**emantic **W**eb **E**nabled **M**ulti-**A**gent **S**ystem

J. Lee and M. Barley (Eds.): PRIMA 2003, LNAI 2891, pp. 122–132, 2003.

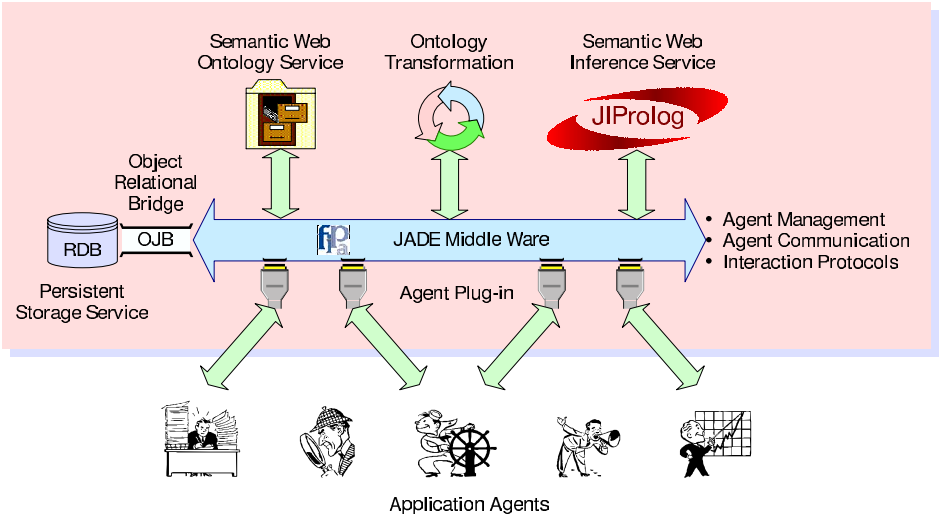


Fig. 1. SweMas: A Multi-Agent Framework Utilizing the Semantic Web

distinguished component to transform between ontologies. These interfaces to the Semantic Web services are built upon the FIPA [6]-compliant JADE agent framework [25]. In our framework JADE serves as a middleware to support agent management, agent communication, and agent interaction protocols. JADE also provides an environment for application agents to be developed and get plugged in to the framework. ObJectRelationalBridge (OJB) is an object-relational mapping tool that allows transparent persistence for Java objects against relational databases [26]. Through OJB, we provide a persistent storage services to maintain persistent objects at agent's requests. For instance, an agent may create a closure of belief and stores it to the persistent storage before the agent asks the inference server to perform an inference task with the closure of belief. In other words, the persistent storage server can works as a shared data server.

In this paper, we first motivate our approach and then lay out the architecture of SweMas to utilize the Semantic Web and the rationales for our design of the framework.

2 Design of the Framework

The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation [2]. Along this objective, several layers of representational structures are proposed as shown in Figure 2.

Our objective is to provide a suitable level of abstraction and functionality for multi-agent systems to tackle the complexity rising as the Semantic Web is enabled in the systems. SweMas aims to incorporate the ontology and logic layers of Semantic Web into multi-agent systems to exploit agent technologies in addition to the Semantic Web. SweMas has been designed with two things in mind: 1. practical open systems; 2. taking

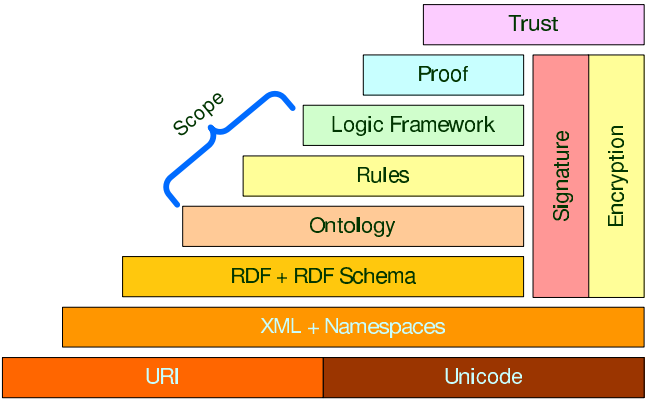


Fig. 2. Semantic Web Layer Stack [9]

advantage of pre-existing tools not to re-invent the wheel. In other words, our approach is focused on integrating existing tools and methodologies to build a practical open multi-agent framework. In the following section, we introduce each component of SweMas and describe the functionality and related issues.

2.1 JADE Middleware

JADE (Java Agent Development Environment) is a software framework to make easy the development of agent applications in compliance with the FIPA specifications for interoperable intelligent multi-agent systems [25]. In our framework JADE serves as a middleware to support agent management, agent communication, and agent interaction protocols. The agent model of JADE is more *primitive* than the agent models offered by other systems, but such primitiveness allows us to have a flexible agent framework to be built on the top of it. JADE offers the following list of features for our framework:

- FIPA-compliant agent platform, which includes the Agent Management System (AMS), the Directory Facilitator (DF), and the Agent Communication Channel (ACC).
- Distributed agent platform
- Transport mechanism and interface to message passing among agents
- Library of FIPA interaction protocols
- Automatic registration of agents with the Agent Management System
- FIPA-compliant naming service
- Graphical user interface to manage several agents and agent platforms from the same agent

The employment of the JADE middleware enables us to jump into the right agent level of abstraction, not worrying about low level system heterogeneousness and interoperability.

2.2 Persistent Storage Service

ObJectRelationalBridge (OJB) [26] is an Object/Relational mapping tool that allows transparent persistence for Java Objects against relational databases. The functionality of OJB listed below allows us to build and integrate a persistent storage service for agents and other Semantic Web services into the SweMas framework.

- OJB uses an XML based Object/Relational Mapping. The mapping resides in a dynamic MetaData layer which can be manipulated at runtime through a simple Meta-Object-Protocol (MOP) to change the behavior of the persistence kernel.
- OJB provides several advanced Object/Relational features like an object caching, lazy materialization through virtual proxies or a distributed lock-management with configurable transaction-isolation levels. Optimistic and pessimistic locking is supported.
- OJB provides a flexible configuration and plug-in mechanism that allows to select from set of predefined components or to implement our own extensions and plug-ins.

Application agents can utilize the Persistent Storage Service to store their internal status such as the world model and intermediate execution results. The Persistent Storage Server can work as a shared data server as well. For example, an agent may create a closure of belief and stores it to the persistent shared storage to request the inference server to perform an inference task with it. SweMas provides a set of well-defined persistence performatives to utilize the Persistent Storage Service and authentication mechanism to protect the information against illegal usages. Persistent Storage Service is specifically indispensable when the ontologies are transformed by Ontology Transformation Service. The transformed ontologies are stored to the Persistent Storage Service and served by the Semantic Web Ontology Service.

2.3 Inference Server

Inference engines are needed to process the knowledge available in the Semantic Web by deducing new knowledge from already specified knowledge [20]. One approach to building an inference engine is to use Problem Solving Methods which are specialized algorithms that specify inference actions to carry out for achieving the goal of a task and define the data flow and the control flow between subtasks as in the IBROW project [1, 12]. Our approach is to have a general logic based inference engine instead. Logic based approaches can be further classified according to the kinds of representation languages as follows:

Description Logic. Description logics are knowledge representation languages tailored for expressing knowledge about concepts and concept hierarchies using a restricted set of first order formulas [15,16]. Knowledge is represented by characterizing classes (or concepts) of objects and the relationship (or links, or roles) among them. The organization of the classes used to describe a domain of interest is based on a hierarchical structure, which not only provides an effective and compact representation of information, but also allows for performing the relevant reasoning tasks in a computationally effective way.

The main reasoning tasks are classification and satisfiability, subsumption and instance checking. Subsumption represents the is-a relation. Classification is the computation of a concept hierarchy based on subsumption. That means, given formulae describing a class, the classifier associated with a certain description logic will place them inside a hierarchy, and given an instance description, the classifier will determine the most specific classes to which the particular instance belongs. A number of description logic implementations are available including FaCT, Grail, Loom, and CLASSIC [10].

Higher Order Logics. Higher Order Logics have the greatest expressive power among all known logics but they lack the completeness property. Higher-order syntax means that variables are allowed to appear in places where normally predicate or function symbols can. In contrast, higher-order semantics has semantic structures in which variables may range over domains of relations and functions constructed out of the domains of individuals.

Available Inference Engines for Higher Order Logics include the generic Theorem Prover Isabelle or HOL [17]. HOL 4 [18] is the latest version of the HOL as a programming environment in which theorems can be proved and proof tools implemented. Built-in decision procedures and theorem provers can automatically establish many simple theorems. HOL 4 is particularly suitable as a platform for implementing combinations of deduction, execution and property checking.

Full First Order Logic. In a first-order semantics, variables can only range over domains of individuals or over the names of predicates and functions, not over sets as such. Predicate calculus is the primary example of a logic where syntax and semantics are both first-order. However, using full first order logic (FOL) for specifying axioms requires a full-fledged automated theorem prover. FOL is semi-decidable and doing inferencing is computationally not tractable for large amounts of data and axioms. Full FOL thus would not scale up for handling huge amounts of knowledge in an environment like the web. Besides full first theorem proving would mean to maintain consistency throughout the web, which is impossible. SETHEO (SEquential THEOrem prover), Otter, SPASS, Gandalf, and TPS (Theorem Proving System) are available automated first order theorem provers [20].

Horn-Logic. Horn-logic is another fragment of First Order Predicate logic. Horn clauses are rules or implicational constraint of formulas of the form $\sigma_1 \wedge \sigma_2 \wedge \ldots \wedge \sigma_n \Rightarrow \rho$, in which the σ_i's and ρ are positive atomic statements. No negation is allowed in either the antecedents or the consequents. Horn clauses form the basis for both the logic programming language Prolog [23] and the database query language Datalog(Horn-logic only with 0-ary function symbols) [3] and a number of efficient evaluation strategies are available for this fragment of predicate logic. Examples of base systems include the XSB system, TRIPLE, and JIProlog.

XSB is an open source logic programming system that extends Prolog with new semantic and operational features, mostly based on the use of Tabled Logic Programming or tabling [30]. TRIPLE, the successor of SiLRI, is an RDF query, inference, and transformation language for the Semantic Web [22,21]. JIProlog (Java Internet Prolog) [4] is a

cross-platform Java Prolog interpreter that supplies Java applications with the features of Prolog language and provides Prolog language with a way to implement new predicates in Java. As a result, JIProlog allows us to call Prolog predicates from Java directly and allows to invoke Java methods from Prolog as predicates are called. JIProlog provides also a set of Prolog predicates to manage XML document using a DOM-like approach.

In the current version of SweMas, we choose to use Horn-logic, particularly Prolog, because of it's simplicity and availability. In the future version, we are planning to add support for the XSB. Prolog is a high level programming language that allows to implement complex systems as a set of logical axioms and inference rules. Unlike procedural languages such Java or C++ where users describe the way to solve a problem by specifying all steps to do, in Prolog users only describe the problem leaving to the interpreter the construction of the way to solve it.

JIProlog [4] has the basic features needed as the basis for the inference service for the Semantic Web. JIProlog basically provides a set of Java classes to link Prolog and Java applications and has the following required features.

- JIProlog can run as a server to provide the Prolog service over the network as well as a stand-alone interpreter. SweMas provides JADE interaction protocols for the JADE agents to request inference services and to get inference results (see Section 2.1).
- The extension package of JIProlog provides a set of Prolog predicates implementing a DOM-like API for parsing and creating XML documents as well as HTML documents. Each DOM object is mapped to a corresponding predicate and has a set of rules to manage it. This feature is handy for the task of interpreting the documents in Semantic Web.
- Another extension package provides a set of Prolog predicates that maps the functionalities of the Java Reflection API. The package allows creating Java objects, invoking their methods, retrieving values from their fields and so on. We use this feature extensively to provide interfaces to the JADE middleware.

2.4 Application Agents

Autonomous agents in the Semantic Web allows automation with minimal or no intervention from programmers. The effectiveness of such autonomous agents increases as the agent can understand more Web contents and can work together with other agents.

Application agents in SweMas can be implemented as any FIPA-compliant JADE agent. SweMas, however, currently provides an interface to only JAM [11] agent. JAM is a Java implementation of the conceptual framework provided by the Procedural Reasoning System (PRS) [8]. JAM maintains a library of alternative procedures for accomplishing goals and selectively invokes and elaborates procedures based on the evolving context. In contrast to the traditional deliberative planning systems, JAM continuously tests its decisions against its changing knowledge about the world, and can redirect the choices of actions dynamically while remaining purposeful to the extent allowed by the unexpected changes to the environment.

The system interacts with its environment, including other systems, through its database (which acquires new beliefs in response to changes in the environment) and through the actions that it performs as it carries out its selected plans.

As a reactive PRS [8] agent architecture, JAM has the following features that are needed in most multi-agent systems.

Real-time execution: The agent is responsive to its environment and predictably fast enough to act on the changes in the environment.

Interruptible execution: The agent is able to pause its current activity and gracefully switch to other more urgent or higher priority activities.

Multiple foci of attention: The agent needs to maintain multiple foci of attention to interact with multiple agents and the environment simultaneously. This ability requires that an agent's activity be not only *interruptible*, but also *resumable*.

Hierarchical plan refinement and revision: The agent can progressively decompose high-level tasks into smaller subtasks. High-level tasks are considered independent of lower level tasks and the low-level tasks are chosen based on the current high-level goals and the current situation.

Purposeful behavior: The agent works methodically toward its goals in a coherent manner.

Adherence to predefined strategies: The agent behaves in a way that is consistent with what others expect.

Goal-driven and data-driven behavior: Agents may be goal-driven, where the agent is motivated by high-level statements about a state of the world to achieve or activities to perform. Agents may also be data-driven, where their behavior is motivated by low-level, perhaps transient data states, such as proximity to danger.

In addition to the features mentioned above, intelligent behavior requires a deliberative inferences. In the next section, we discuss the issue of incorporating the deliberative inferences into the reactive JAM agent architecture.

2.5 Deliberative Inferences for Reactive Agents

JAM agents are reactive and thus responsive to their environment. They are expected to be fast enough to act on the changes in the environment. On the other hand, inference processes for intelligent behaviors are inevitably time consuming compared to the reaction cycle of reactive agents. Balancing of fast reactions and deliberation of long-term ramifications such the inference processes is an important problem [13]. In this section, we introduce our approach to minimizing the need for inference processes by cutting down unnecessary triggers for inferences. First we describe how reactive behaviors are represented in JAM and then explain our approach.

Representation of Reactive Behaviors. A plan in JAM defines a procedural specification for accomplishing a goal. Its applicability is limited to a particular goal and to a certain precondition and context. The *precondition* specifies the initial conditions that must be met before the plan should be considered for execution. The precondition thus is checked only when a plan is being considered for selection but not checked during runtime.

On the other hand, a plan's *context* specifies the conditions under which the plan will be useful throughout the duration of plan execution. The context of a plan is first checked

when a plan is being considered for selection and checked throughout execution of a plan to make sure that the plan is still applicable even in the face of a changing environment. If the context of a currently executing plan fails, the entire plan and any of its subgoals are deemed invalid and aborted.

The context of a plan plays the role of making agent's behaviors reactive because it is continuously monitored throughout the execution of the plan. However such a continuous checking can be costly if the conditions in the plan's context include some results of inferences because a repeated testing of the condition would cause repeated inference requests. If we can *cache* the truth of the inferenced results and the cached results are properly updated as the supporting facts for the inferenced results are changed, the costly unnecessary inferences would be saved. The key to success for caching is the proper representation of the influence or causal relation from the supporting facts to the conditions. We employ the Rete algorithm [5] to represent this causal relation and to cache the inferenced results.

Rete algorithm. Charles Forgy created the original Rete algorithm around 1982 as part of his DARPA-funded research. Compared to many previous production-matching algorithms, Rete was very advanced. Even today, there have been few improvements to it in the general case. Variations on Rete, such as TREAT, may have different performance characteristics depending on the environment. Some perform better with large rule sets but small numbers of objects, while other perform well for steady-state environments, but react poorly to numerous successive changes in the data [27].

A Rete network is a graph through which data flows. Originally, data was specified using Cambridge-prefix tuples since Lisp-like languages were in style for logic programming. The tuples were used to express attributes about objects. The tuples are dropped into the Rete network, and those that reach the far end cause the ring of a rule. The original production-matching was based upon matches against tuple patterns.

3 Applications

We are currently working to apply the SweMas framework to implement a Multidisciplinary Design Optimization (MDO) framework. MDO is an optimization technique considering simultaneously multiple disciplines such as dynamics, mechanics, structural analysis, thermal and fluid analysis and electromagnetic analysis. A software system enabling multidisciplinary design optimization is called MDO framework. An MDO framework provides an integrated and automated design environment that increases product quality and reliability, and decreases design cycle time and cost. The MDO framework also works as a common collaborative workspace for design experts on multiple disciplines.

In this application, we has performed the requirement analysis through interviews of design experts in industry to reflect the real needs in industry. The requirements include integrated design environment, friendly user interface, highly extensible open architecture, distributed design environment, application program interface, and efficient data management to handle massive design data.

The resultant MDO framework is a multi-agent framework where each agent represents the specialized disciplines. In this approach, each discipline has its own ontology encapsulating the domain knowledge. This knowledge includes how to interpret the output result of the tools that are used for the discipline and how to interact with other agents. Agents that were not explicitly designed to work together can coordinate their activities by exchanging their ontologies. As a result, we can build an extensible design framework with effective data exchanges and interactions in the distributed environment of multiple design tools and software.

4 Conclusions and Future Work

We have described the architecture of our multi-agent framework called SweMas that utilizes the Semantic Web and presented the rationales for our design of the framework. SweMas offers a development environment, new capabilities, and operating environment. We have implemented SweMas as a practical open system using Java programming language taking advantage of pre-existing tools. SweMas is being applied to multidisciplinary design optimization problems and tested for usability and adaptability. For a software framework such as our SweMas, it is hard to compare with other systems based on quantitative measures. Nonetheless we believe that our framework lays a stepping stone for a foundation of Semantic Web enabled multi-agent systems. Qualitatively speaking, SweMas's compliance to FIPA standards offers openness and adaptability for the future extensions. SweMas also provides a suitable level of abstraction and functionality for multi-agent systems to tackle the complexity rising as the Semantic Web is enabled in the systems.

We have to admit that SweMas is the first step toward our goal toward a practical multi-agent framework utilizing the Semantic Web. In particular, the architectural component for ontology transformation is in the primitive development stage. As part of future work, we intend to explore and support explicit methods for ontology transformation between OWL and Topic Maps [24] because of its imminent needs in industry [7,14, 19]. We wish that our framework serves as a practical guideline for the implementation of multi-agent systems.

Acknowledgment. This work was supported by the Korea Science and Engineering Foundation (KOSEF) through the Advanced Information Technology Research Center (AITrc) and Korea Science and Engineering Foundation (KOSEF) through the Center of Innovative Design Optimization Technology (iDOT).

References

1. V. Richard Benjamins, Enric Plaza, Enrico Motta, Dieter Fensel, Rudi Studer, Bob Wielinga, Guus Schreiber, and Zdenek Zdrahal. IBROW3 - an intelligent brokering service for knowledge-component reuse on the world wide web. In *Proceedings of the Eleventh Workshop on Knowledge Acquisition, Modeling and Management (KAW'98)*, Banff, Alberta, Canada, April 1998.

2. Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web: A new form of web content that is meaningful to computers will unleash a revolution of new possibilities. *Scientific American*, 284(5):34–43, May 2001.
3. S. Ceri, G. Gottlob, and L. Tanca. *Logic Programming and Databases*. Springer-Verlag, 1990.
4. Ugo Chirico. JIProlog - Java Internet Prolog. `http://www.ugosweb.com/jiprolog/`, July 2003.
5. Charles Forgy. Rete: A fast algorithm for the many patterns/many objects match problem. *Artificial Intelligence*, 19(1):17–37, 1982.
6. Foundation for Intelligent Physical Agents. Fipa specifications. `http://www.fipa.org/specifications/`, 2002.
7. Lars Marius Garshol. Living with topic maps and RDF. `http://www.ontopia.net/topicmaps/materials/tmrdf.html`, 2003.
8. Michael P. Georgeff and François Félix Ingrand. Decision-making in an embedded reasoning system. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 972–978, Detroit, Michigan, 1989.
9. James Hendler. Agents and the semantic web. *IEEE Intelligent Systems*, 16(2):30–37, March/April 2001.
10. Ian Horrocks. Description logics. `http://www.cs.man.ac.uk/ai/Software/description_logics.html`, 2003.
11. Marcus Huber. JAM: A BDI-theoretic mobile agent. In *Proceedings of the Third International Conference on Autonomous Agents (Agents '99)*, Seattle, Washington, May 1999.
12. Information Society Technologies. An intelligent brokering service for knowledge-component reuse on the world-wide web. `http://www.swi.psy.uva.nl/projects/IBROW3/home.html`, February 2000.
13. Russell Knight, Forest Fisher, Tara Estlin, Barbara Engelhardt, and Steve Chien. Balancing deliberation and reaction, planning and execution for space robotic applications. In *Proceedings of the 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2131–2139, Maui, Hawaii, USA, October/November 2001.
14. Martin S. Lacher and Stefan Decker. On the integration of topic maps and rdf data. In *Proceedings of the First Semantic Web Workshop*, July 2001.
15. Patrick Lambrix. Description logics. http://www.ida.liu.se/labs/iislab/people/patla/DL/, 2003.
16. Carsten Lutz. Description logics. `http://dl.kr.org`, 2003.
17. Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel. *Isabelle/HOL — A Proof Assistant for Higher-Order Logic*, volume 2283 of *Lecture Notes in Computer Science*. Springer, 2002.
18. Michael Norrish. HOL 4. `http://hol.sourceforge.net/`, SourceForge.net.
19. Steve Pepper. Ten theses on topic maps and rdf. `http://www.ontopia.net/topicmaps/materials/rdf.html`, October 2002.
20. The Semantic Web Community Portal. Inference engines for the semantic web. `http://www.semanticweb.org/inference.html`, 2003.
21. Michael Sintek and Stefan Decker. TRIPLE. `http://triple.semanticweb.org/`, The Semantic Web Foundation for Open Source Software (SFO), 2002.
22. Michael Sintek and Stefan Decker. TRIPLE — a query, inference, and transformation language for the semantic web. In Michael Sintek and Stefan Decker, editors, *Proceedings of the First International Semantic Web Conference (ISWC-2002)*, Sardinia, Italy, June 2002.
23. L. Sterling and E. Shapiro. *The Art of Prolog*. MIT Press, 1986.
24. XML topic maps (XTM) 1.0, TopicMaps.Org specification. `http://www.topicmaps.org/xtm/1.0/`, 2003.
25. Telecom Italia Lab. Java agent development framework. `http://sharon.cselt.it/projects/jade/`, March 2003.

26. The Apache DB Project. ObJectRelationalBridge. http://db.apache.org/ojb, 2003.
27. The Werken Company. The drools guide. http://drools.org/pdf/drools-guide.pdf, January 2003.
28. Raphael Volz, Stefan Decker, and Daniel Oberie. Bubo - implementing OWL in rule-based systems. http://www.daml.org/listarchive/joint-committee/att-1254/01-bubo.pdf, May 2003.
29. W3C. Owl web ontology language overview. http://www.w3.org/TR/owl-features/, March 2003.
30. XSB development team. XSB. http://xsb.sourceforge.net, Computer Science Department of the Stony Brook University, January 2000.

Speculative Constraint Processing in Multi-agent Systems

Ken Satoh[1], Philippe Codognet[2], and Hiroshi Hosobe[1]

[1] National Institute of Informatics
2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430, Japan,
{ksatoh,hosobe}@nii.ac.jp
[2] University of Paris 6, LIP6/IA,
8 rue du Capitaine Scott, 75015 Paris, France
Philippe.Codognet@lip6.fr

Abstract. In this paper, we extend our framework of speculative computation in multi-agent systems by using default constraints. In research on multi-agent systems, handling incomplete information due to communication failure or due to other agents' delay in communication, is a very important issue. For a solution to this problem, we previously proposed *speculative computation* based on *abduction* in the context of master-slave multi-agent systems and gave a procedure in abductive logic programming. In the proposal, a master agent prepares a default value for a yes/no question in advance and it performs speculative computation using the default without waiting for a reply to the question. This computation is effective unless the contradictory reply to the default is returned. In this paper, we formalize speculative constraint processing and propose a correct procedure for such computation so that we can handle not only yes/no questions, but also more general types of questions.

1 Introduction

In this paper, we extend a method of *speculative computation* in master-slave multi-agent systems [Satoh00] by using constraints in order to manipulate various types of questions instead of yes/no questions.

In most of the current research on multi-agent systems, people assume that communication of agents is guaranteed. Also, when an agent asks a question of other agents, a process is suspended until some response from other agents is obtained. However, in real settings such as the Internet, this assumption is not guaranteed. Moreover, even if communication is guaranteed, when an agent needs to communicate with other agents and the computation in other agents takes much time before sending an answer, we encounter a similar situation.

For problem solving in the above situations, we proposed speculative computation [Satoh00] based on *abduction*. When communication is delayed or failed, then we use a default hypothesis as a tentative answer and continue a computation. When some response is obtained, we check the consistency of the response

J. Lee and M. Barley (Eds.): PRIMA 2003, LNAI 2891, pp. 133–144, 2003.

and the default. If the response is consistent, then we continue the current computation; else if the response is inconsistent, we perform an alternative computation. This is desirable in the situation where some action should be taken in advance although a complete plan cannot be decided because of incompleteness of information.

In [Satoh00], we proposed an implementation of speculative computation for a master-slave multi-agent system and showed that abduction plays an important role in speculative computation.

The method proposed in [Satoh00] only considers yes/no questions for speculative computation. However, in more general settings, an agent asks another agent a question on unknown values.

As an example of speculative computation, consider the following meeting room reservation.

- There are three persons A, B and C to attend a meeting during days 1, 2 and 3.
- If a person is available on a day, then he/she will attend the meeting on the day.
- We ask each person on which date he/she is free or not.
- If all the persons are available on the same day, we can reserve a large room for the day.
- If only two persons are available on the same day, we can reserve a small room for the day.

Our task is to make a plan of possible meeting room reservation for days 1, 2 and 3. We assume that there is a considerable difference between a small room and a large room, so we have to choose either of them.

Suppose that we receive an answer from A that A is free on day 1 and day 2 and busy on day 3, and an answer from B that B is free on day 2 and day 3 and busy on day 1, and also suppose that an answer from C is delayed.

If we follow the requirement that the communication must be completed before we take any further action, then, we cannot make any reservation until we get an answer from C. In ordinary life, however, if we know that C is normally free on the specific dates, then we can tentatively make a conclusion. For example, suppose that we know that C is normally free on day 2 and day 3. Then, we could make a reservation of the large room for day 2 if we decide to have a meeting on day 2, or make a reservation of the small room for day 3 if we decide to have a meeting on day 3.

In the previous framework, we can only ask a yes/no question, so we have to make a question of availability for each day. However, it is more efficient if we ask available days in one question. To solve the problem, we introduce *constraints* into our framework so that we can ask other agents about possible values or constraints of questions (such as possible available days in the previous example).

The basic idea of this method is as follows.

1. The agent A prepares a default constraint for variables in a question in advance.

2. When an agent A asks a question of another agent B, the agent A uses the default constraint as a tentative answer and continues a computation along with the tentative answer.
3. When the response comes from the agent B,
   - if the response entails the default answer, then the agent A continues the computation.
   - if the response is inconsistent with the default answer, the agent A withdraws the computation process which uses the default answer. Then A restarts a computation with the true answer.
   - if the response does not entail the default answer, but is consistent with the default answer, A continues the computation which uses the default, and simultaneously A starts alternative computation as well.

We assume that the default answer is prepared to cover the normal answer of each agent. This means that the response usually entails the default answer. We, therefore, expect that the computation which uses the default answer is usually effective.

Unlike a yes/no question, however, we must consider the above third case. In yes/no questions, the response either entails the default or is inconsistent with the default. On the other hand, in our extended setting, there is a possibility that the response neither entails the default nor contradicts with the default. In this case, we must consider both of the computations one of which uses the default and the other of which does not use the default.

In this paper, we restrict our attention to a master-slave multi-agent system where the master uses speculative computation. We propose an implementation of a speculative framework using constraints and prove the correctness of this method. Then, we show an effect of speculative computation by an example.

2 Framework of Constraint Processing in Master-Slave System

In this section, we firstly provide a framework of constraint handling in a master-slave system. The framework follows the CLP (Constraint Logic Programming) framework.

Definition 1. *A* constraint framework for a master-slave system *is a triple* $\langle \Sigma, \mathcal{E}, \mathcal{P} \rangle$ *where*

- Σ *is a finite set of constants. We call an element in* Σ *a* slave agent identifier.
- $\mathcal{E}$ *is a set of predicates called* external predicates. *When* Q *is an atom with an external predicate and* S *is a slave agent identifier, we call* $Q@S$ *an* askable atom.
- $\mathcal{P}$ *is a constraint logic program, that is, a set of rules of the form: "*$H \leftarrow C \| B_1, B_2, ..., B_n$*." where*
  - H *is an atom with a non-external predicate called* head *of the above rule* R *denoted as* $head(R)$.

- *C is a set of constraints called* body constraints *of R denoted as $const(R)$.*
- *each of $B_1, ..., B_n$ is either an atom or an askable atom and we refer to $B_1, ..., B_n$ as* body *of R denoted as $body(R)$.*

Example 1. The example of meeting room reservation in Introduction (without speculative computation) is represented as the following framework $\langle \Sigma, \mathcal{E}, \mathcal{P} \rangle$ with set constraints.

- $\Sigma = \{a, b, c\}$
- $\mathcal{E} = \{free, busy\}$
- $\mathcal{P}$ is the following set of rules[1]:
  $plan(small\_room, [X, Y], D) \leftarrow D \in \{1, 2, 3\} \| meeting([X, Y], D).$
  $plan(large\_room, [a, b, c], D) \leftarrow D \in \{1, 2, 3\} \| meeting([a, b, c], D).$
  $meeting([a, b], D) \leftarrow \| available(a, D), available(b, D), non\_available(c, D).$
  $meeting([b, c], D) \leftarrow \| non\_available(a, D), available(b, D), available(c, D).$
  $meeting([c, a], D) \leftarrow \| available(a, D), non\_available(b, D), available(c, D).$
  $meeting([a, b, c], D) \leftarrow \| available(a, D), available(b, D), available(c, D).$
  $available(P, D) \leftarrow \| free(D)@P.$
  $non\_available(P, D) \leftarrow \| busy(D)@P.$

The execution of the above framework is defined as follows which is straightforwardly adapted from the execution of the usual CLP framework. For a non-askable atom in a goal, we reduce it into subgoals by the above rule and for an askable atom in a goal, a master agent asks a slave agent a query with the current constraints and waits for the answer. The answer is returned as a set of constraints on variables in the query. When the answer constraints are returned, they are added into the current constraints and the execution is resumed. The execution is completed when the empty goal is found.

Definition 2. *A goal is of the form $\leftarrow C \| B_1, ..., B_n$ where*

- *C is a set of constraints.*
- *each of $B_1, ..., B_n$ is either an atom or an askable atom.*

For a semantics of the above framework, we handle the semantics of an askable atom as if we knew the reply for a question in the askable atom beforehand.

Definition 3. *A reply set for $\mathcal{E}$ is a set of rules of the form: $Q@S \leftarrow C \| .$, where $Q@S$ is an askable atom and each argument of Q is a variable and C is a set of constraints over those variables.*

Note that a reply set is not necessarily specified for every askable atom. If a reply set is not specified for an askable atom, we regard an answer for the askable atom as undecided.

[1] A string beginning with an upper case letter represents a variable and a string beginning with a lower case letter represents a constant.

Definition 4. *A reduction of a goal* $\leftarrow C \| B_1, ..., B_n$ *w.r.t. a constraint logic program* $\mathcal{P}$ *and a reply set* $\mathcal{R}$ *and a subgoal* B_i *is a goal* $\leftarrow C' \| B'$ *such that*

- *there is a rule* R *in* $\mathcal{P} \cup \mathcal{R}$ *s.t.* $C \wedge \{B_i = head(R)\} \wedge const(R)$ *is consistent and* $\{B_i = head(R)\}$ *is a conjunction of constraints equating the arguments of atoms* B_i *and* $head(R)$.
- $C' = C \wedge \{B_i = head(R)\} \wedge const(R)$
- $B' = \{B_1, ...B_{i-1}, B_{i+1}, ..., B_n\} \cup body(R)$

Definition 5. *A derivation of a goal* $\leftarrow C \| B_1, ..., B_n$ *w.r.t. a constraint framework* $\langle \Sigma, \mathcal{E}, \mathcal{P} \rangle$ *and a reply set* $\mathcal{R}$ *is a sequence of reductions* $\leftarrow C \| B_1, ..., B_n, ..., \leftarrow C' \| \emptyset$ *w.r.t.* $\mathcal{P}$ *and* $\mathcal{R}$ *where* $\emptyset$ *denotes an empty goal.* C' *is called an* answer constraint *w.r.t. the goal, the framework and the reply set.*

Example 2. Suppose that a reply set is the following:

$free(D)@a \leftarrow D \in \{1\} \| .$, $busy(D)@a \leftarrow D \in \{2, 3\} \| .$
$free(D)@b \leftarrow D \in \{2, 3\} \| .$, $free(D)@c \leftarrow D \in \{1, 3\} \| .$
$busy(D)@c \leftarrow D \in \{2\} \| .$

Then, we have $R = small\_room, L = [X, Y], X = b, Y = c, D \in \{3\}$ as an answer constraint w.r.t. a goal $\leftarrow \emptyset \| plan(R, L, D)$, and the above framework and the above reply set. Note that we cannot conclude $R = small\_room, L = [X, Y], X = a, Y = c, D \in \{1\}$ since there is no information on $busy(D)@b$.

3 Speculative Framework in Master-Slave System

Now we extend the constraint framework to perform a speculative computation.

Definition 6. *A* speculative framework for a master-slave system SF_{MS} *is a quadruple* $\langle \Sigma, \mathcal{E}, \Delta, \mathcal{P} \rangle$ *where*

- Σ, $\mathcal{E}$ *and* $\mathcal{P}$ *are the same as above in the constraint framework.*
- Δ*: a set of rules of the following form called* default rule *w.r.t.* $Q@S$*:*

$$Q@S \leftarrow C \| .$$

where $Q@S$ *is an askable atom and* C *is a set of constraints called* default constraint for $Q@S$*. We denote a default rule w.r.t.* $Q@S$ *as* $\delta(Q@S)$.

In the above definition an askable atom is used for two purposes. One is for a question sent by a master agent to a slave agent and the other is for a specification of default constraints. If there is no replies returned already for an askable atom, we use a default constraint for the askable atom as a tentative answer from the other agents.

Example 3. The example of meeting room reservation in Introduction is represented as the following speculative framework $SF_{MS} = \langle \Sigma, \mathcal{E}, \Delta, \mathcal{P} \rangle$ with set constraints where a is expected to be free on days 1 and 2 and busy on day 3, b is expected to be free on days 1 and 3, and c is expected to be free on day 3 and busy on day 2.

- Σ, $\mathcal{E}$ and $\mathcal{P}$ are the same as in Example 1.
- Δ is the set of the following rules:
  $free(D)@a \leftarrow D \in \{1,2\}\|.$, $busy(D)@a \leftarrow D \in \{3\}\|.$
  $free(D)@b \leftarrow D \in \{1,3\}\|.$, $free(D)@c \leftarrow D \in \{3\}\|.$
  $busy(D)@c \leftarrow D \in \{2\}\|.$

4 An Operational Model for Speculative Computation

In this section, we propose an abstract implementation of speculative computation. The execution of the speculative framework is based on two phases, a *process reduction phase* and a *fact arrival phase*. The process reduction phase is a normal execution of a program in a master agent and the fact arrival phase is an interruption phase when an answer arrives from a slave agent.

An active process consists of the following objects; (a) the current status of computation including the current constraint set and (b) a set of askable atoms which have been reduced already. Each process represents an alternative way of computation. Intuitively, processes are created when a choice point of computation is encountered such as case splitting or default handling. An active process ends successfully if all the computation is done and the constraints associated with reduced askable atoms have not been contradictory with the current reply set. A process fails when some used default constraints are found to contradict the newly returned answer.

In the process reduction phase, we reduce an active process set (see Definition 9 for the formal definition) into a new process set. During the reduction, for a process using a default constraint, the default is assumed unless an inconsistent constraint with the default has already been assumed or found to be true, whereas a process will be killed when the default used in the process contradicts the current constraint. When we start a speculative computation using the default value, we create another process to keep alternative computation which does not use the default value. We suspend this alternative process since the probability of failure of the alternate process is high.

When an answer comes from a slave agent, we consider the following three possibilities:

- If the answer entails the default, we continue the process using the default and remove the alternative suspended process which was created when a speculative computation starts.
- If the answer contradicts the default, we remove processes using the default and resume the alternative suspended processes.
- If the answer does not entail the default, but is consistent with the default, we not only continue the process using the default by adding the returned answer, but also resume the alternative process.

4.1 Preliminary Definitions

We define the following objects for process reduction.

Definition 7. *An* active process *is a pair* $\langle \leftarrow C \| GS, AD \rangle$ *which consists of:*
- $\leftarrow C \| GS$*: A goal consisting of a set of atoms* GS *and a set of constraints* C*.*
- AD*: A set of askable atoms assumed already called* assumed askable atoms*.*

Definition 8. *A* suspended process *is a triple* $\langle SG, \leftarrow C \| GS, AD \rangle$ *which consists of:*
- SG*: An askable atom called a* suspended atom*.*
- $\leftarrow C \| GS$*: A goal.*
- AD*: A set of assumed askable atoms.*

We use the following four sets for process reduction.

Definition 9.
- An active process set APS *is a set of active processes.*
- A suspended process set SPS *is a set of suspended processes.*
- Already asked queries AAQ *are a set of askable atoms.*
- Returned facts RF *are a set of rules of the form:* $Q@S \leftarrow C \|$ *where* $Q@S$ *is an askable atom and* C *is a set of constraints.*

AAQ is used to avoid asking redundant questions of slave agents, and RF is a set of true constraints returned from slave agents about askable atoms of the form $Q@S \leftarrow C \|$.

4.2 Process Reduction Phase

In the following reduction, we specify changed APS, SPS, AAQ, RF as $NewAPS$, $NewSPS$, $NewAAQ$, $NewRF$; otherwise each of APS, SPS, AAQ, RF is unchanged.

Initial Step: Let GS be an initial goal set. We give $\langle \leftarrow \emptyset \| GS, \emptyset \rangle$ to a proof procedure. That is, $APS = \{\langle \leftarrow \emptyset \| GS, \emptyset \rangle\}$. And let $SPS = AAQ = RF = \emptyset$.

Iteration Step: Do the following.

Case 1. If there is an active process $\langle \leftarrow C \| \emptyset, AD \rangle$, then output constraints C and a set of assumed askable atoms which is in AD which is not in $HEAD(RF)$ where $HEAD(RF) = \{head(R) | R \in RF\}$.

Case 2. Otherwise, select an active process $\langle \leftarrow C \| GS, AD \rangle$ from APS and select an atom L in GS. Let $APS' = APS - \{\langle \leftarrow C \| GS, AD \rangle\}$ and $GS' = GS - \{L\}$.

For the selected atom L, do the following.

- If L is a non-askable atom,
  $NewAPS = APS' \cup$
  $\{\langle \leftarrow (C \wedge \{B_i = head(R)\} \wedge const(R)) \| (body(R) \cup GS'), AD \rangle |$
  $C \wedge \{B_i = head(R)\} \wedge const(R)$ is consistent.$\}$
- If L is an askable atom, $Q@S$, then
  - if $L \notin AAQ$, then send a question Q to a slave agent S and $NewAAQ = AAQ \cup \{L\}$.
  - if $L \in AD$ then $NewAPS = APS' \cup \{\langle \leftarrow C \| GS', AD \rangle\}$

- else if $(L \leftarrow C_r \|) \in RF$ then
  if $C \wedge C_r$ is consistent then
  $NewAPS = APS' \cup \{\langle \leftarrow C \wedge C_r \| GS', AD \rangle\}$
  else $NewAPS = APS'$
- else if a default constraint C_d for L exists then,
  * if $C \wedge C_d$ is consistent then
    $NewAPS = APS' \cup \{\langle \leftarrow C \wedge C_d \| GS', AD \cup \{L\} \rangle\}$
    else $NewAPS = APS'$
  * if $C \wedge \neg C_d$ is consistent then
    $NewSPS = SPS \cup \{\langle L, \leftarrow C \wedge \alpha \| GS', AD \rangle\}$ where $C \wedge \neg C_d \models \alpha$.

If the constraint solver can manipulate any logical combination of constraints such as negations and disjunctions, the above α can be equivalent to $C \wedge \neg C_d$. If the solver cannot do so, α might be a partial constraint weaker than $C \wedge \neg C_d$ or sometimes no constraint.

4.3 Fact Arrival Phase

Suppose that a constraint is returned from a slave agent S for a question, $Q@S$. We denote the returned constraint as $Q@S \leftarrow C_r \|$. Then, do the following after one step of process reduction is finished.

- $NewRF = RF \cup \{Q@S \leftarrow C_r \| \}$
- If a default constraint C_d for $Q@S$ exists then,
  - If C_r entails C_d,
    * $NewAPS = APS - DeletedAPS \cup AddedAPS$
      where $DeltedAPS = \{\langle \leftarrow C \| GS, AD \rangle \in APS | Q@S \in AD\}$
      and $AddedAPS = \{\langle \leftarrow (C \wedge C_r) \| GS, AD \rangle |$
      $\langle \leftarrow C \| GS, AD \rangle \in DeletedAPS$ and $C \wedge C_r$ is consistent$\}$
    * $NewSPS = SPS - DeletedSPS \cup AddedSPS$
      where $DeletedSPS = \{\langle SG, \leftarrow C \| GS, AD \rangle \in SPS |$
      $SG = Q@S$ or $Q@S \in AD\}$
      and $AddedSPS = \{\langle SG, \leftarrow (C \wedge C_r) \| GS, AD \rangle |$
      $\langle SG, \leftarrow C \| GS, AD \rangle \in DeletedSPS$
      and $Q@S \in AD$ and $C \wedge C_r$ is consistent$\}$
  - If C_r contradicts C_d,
    * $NewAPS = APS - DeletedAPS \cup ResumedSPS$
      where $DeltedAPS = \{\langle \leftarrow C \| GS, AD \rangle \in APS | Q@S \in AD\}$
      and $ResumedSPS = \{\langle \leftarrow (C \wedge C_r) \| GS, AD \rangle |$
      $\langle Q@S, \leftarrow C \| GS, AD \rangle \in SPS$ and $C \wedge C_r$ is consistent$\}$
    * $NewSPS = SPS - DeletedSPS$
      where $DeletedSPS = \{\langle SG, \leftarrow C \| GS, AD \rangle \in SPS |$
      $SG = Q@S$ or $Q@S \in AD\}$
  - If C_r does not entail C_d nor contradicts C_d,

$*$ $NewAPS = APS - DeletedAPS \cup AddedAPS \cup ResumedSPS$
where $DeltedAPS = \{\langle \leftarrow C \| GS, AD\rangle \in APS | Q@S \in AD\}$
and $AddedAPS = \{\langle \leftarrow (C \wedge C_r) \| GS, AD\rangle |$
$\langle \leftarrow C \| GS, AD\rangle \in DeletedAPS$ and $C \wedge C_r$ is consistent$\}$
and $ResumedSPS = \{\langle \leftarrow (C \wedge C_r) \| GS, AD\rangle |$
$\langle Q@S, \leftarrow C \| GS, AD\rangle \in SPS$ and $C \wedge C_r$ is consistent$\}$
$*$ $NewSPS = SPS - DeletedSPS \cup AddedSPS$
where $DeletedSPS = \{\langle SG, \leftarrow C \| GS, AD\rangle \in SPS |$
$SG = Q@S$ or $Q@S \in AD\}$
and $AddedSPS = \{\langle SG, \leftarrow (C \wedge C_r) \| GS, AD\rangle |$
$\langle SG, \leftarrow C \| GS, AD\rangle \in DeletedSPS$
and $Q@S \in AD$ and $C \wedge C_r$ is consistent$\}$

Note that we can select any goal of any active process for a goal reduction and we guarantee the correctness of this procedure by the theorems below.

4.4 Correctness of the Proof Procedure

The following theorems show correctness and a kind of completeness for the above procedure.

Theorem 1. *Let $SF_{MS} = \langle \Sigma, \mathcal{E}, \Delta, \mathcal{P}\rangle$ be a speculative framework. Let G be an initial goal set and OD be used assumed askable atoms and C be an answer constraint from the above procedure, and RF be a set of constraints returned from the other agents when the output is obtained. Then, there exists an answer constraint C' w.r.t. a constraint framework $\langle \Sigma, \mathcal{E}, \mathcal{P}\rangle$ and a reply set $RF \cup \{\delta(Q@S) | Q@S \in OD\}$ s.t. C entails C'.*

Theorem 2. *Let $SF_{MS} = \langle \Sigma, \mathcal{E}, \Delta, \mathcal{P}\rangle$ be a speculative framework. Suppose that we fix a partial reply set RF and let $\mathcal{R}$ be $RF \cup \{\delta(Q@S) | Q@S \in \Delta$ and $Q@S \notin HEAD(RF)\}$ where $HEAD(RF) = \{head(R) | R \in RF\}$. Let C be an answer constraint w.r.t. a constraint framework $\langle \Sigma, \mathcal{E}, \mathcal{P}\rangle$ and a reply set $\mathcal{R}$. Then, there are some computational paths in the above procedure which output $C_1, ..., C_n$ s.t. $C = C_1 \vee ... \vee C_n$.*

The above theorems mean that answer constraints from an ordinary constraint computation are decomposed into partial ones in a speculative computation.

5 Example

We use the program in Example 3 and take the following strategy for process reduction.

- When we reduce an atom, new processes are created along with the rule order in the program which are unifiable with the atom.
- We always select a newly created or a newly resumed process and a left-most atom.

The following is an execution trace for $plan(R, L, D)$. We assume that answers from the agents b and a come at step 16 and step 17 respectively. We show changes of active processes and suspended processes during the execution.

1. Active: $\langle(\leftarrow \emptyset \| plan(R, L, D)), \emptyset\rangle$
2. Active: $\langle(\leftarrow D \in \{1,2,3\}, R = sr, L = [X, Y] \| plan(sr, [X, Y], D)), \emptyset\rangle$,
   $\langle(\leftarrow D \in \{1,2,3\}, R = lr, L = [a, b, c] \| plan(lr, [a, b, c], D)), \emptyset\rangle$[2]
3. Active: $\langle(\leftarrow D \in \{1,2,3\}, R = sr, L = [X, Y] \| mt([X, Y], D)), \emptyset\rangle$,
   $\langle(\leftarrow D \in \{1,2,3\}, R = lr, L = [a, b, c] \| plan(lr, [a, b, c], D)), \emptyset\rangle$
4. Active: $\langle(\leftarrow D \in \{1,2,3\}, \theta_{ab} \| av(a, D), av(b, D), n\_av(c, D)), \emptyset\rangle$,
   $\langle(\leftarrow D \in \{1,2,3\}, \theta_{bc} \| n\_av(a, D), av(b, D), av(c, D)), \emptyset\rangle$,
   $\langle(\leftarrow D \in \{1,2,3\}, \theta_{ca} \| av(a, D), n\_av(b, D), av(c, D)), \emptyset\rangle$,
   $\langle(\leftarrow D \in \{1,2,3\}, \theta_{abc} \| plan(lr, [a, b, c], D)), \emptyset\rangle$[3]
5. Active: $\langle(\leftarrow D \in \{1,2,3\}, \theta_{ab} \| fr(D)@a, av(b, D), n\_av(c, D)), \emptyset\rangle$,
   P_{bc}, P_{ca}, P_{abc}[4]
6. $fr(D)$ is asked of a and since $(fr(D)@a \leftarrow D \in \{1,2\} \|) \in \Delta$,
   Active: $\langle(\leftarrow D \in \{1,2\}, \theta_{ab} \| av(b, D), n\_av(c, D)), \{fr(D)@a\}\rangle$, P_{bc}, P_{ca}, P_{abc}
   Suspended: $\langle fr(D)@a, (\leftarrow D \in \{3\}, \theta_{ab} \| av(b, D), n\_av(c, D)), \emptyset\rangle$
7. Active: $\langle(\leftarrow D \in \{1,2\}, \theta_{ab} \| fr(D)@b, n\_av(c, D)), \{fr(D)@a\}\rangle$, P_{bc}, P_{ca}, P_{abc}
   Suspended: SP_1[5]
8. $fr(D)$ is asked of b and since $(fr(D)@b \leftarrow D \in \{1,3\} \|) \in \Delta$,
   Active: $\langle(\leftarrow D \in \{1\}, \theta_{ab} \| n\_av(c, D)), \{fr(D)@a, fr(D)@b\}\rangle$, P_{bc}, P_{ca}, P_{abc}
   Suspended: SP_1, $\langle fr(D)@b, (\leftarrow D \in \{2\}, \theta_{ab} \| n\_av(c, D)), \{fr(D)@a\}\rangle$
9. Active: $\langle(\leftarrow D \in \{1\}, \theta_{ab} \| bsy(D)@c), \{fr(D)@a, fr(D)@b\}\rangle$, P_{bc}, P_{ca}, P_{abc}
   Suspended: SP_1, SP_2[6]
10. $bsy(D)$ is asked of c and since $(bsy(D)@c \leftarrow D \in \{2\} \|) \in \Delta$,
    Active: P_{bc}, P_{ca}, P_{abc}
    Suspended: SP_1,SP_2, $\langle bsy(D)@c, (\leftarrow D \in \{1\}, \theta_{ab} \| \emptyset), \{fr(D)@a, fr(D)@b\}\rangle$
11. Active: $\langle(\leftarrow D \in \{1,2,3\}, \theta_{bc} \| bsy(D)@a, av(b, D), av(c, D)), \emptyset\rangle$, P_{ca}, P_{abc}
    Suspended: SP_1,SP_2,SP_3[7]
12. $bsy(D)$ is asked of a and since $(bsy(D)@a \leftarrow D \in \{3\} \|) \in \Delta$,
    Active: $\langle(\leftarrow D \in \{3\}, \theta_{bc} \| av(b, D), av(c, D)), \{bsy(D)@a\}\rangle$, P_{ca}, P_{abc}
    Suspended: SP_1,SP_2,SP_3,$\langle bsy(D)@a, (\leftarrow D \in \{1,2\}, \theta_{bc} \| av(b, D), av(c, D)), \emptyset\rangle$
13. Active: $\langle(\leftarrow D \in \{3\}, \theta_{bc} \| fr(D)@b, av(c, D)), \{bsy(D)@a\}\rangle$, P_{ca}, P_{abc}
    Suspended: SP_1,SP_2,SP_3,SP_4[8]

---

[2] We abbreviate *small_room* as sr, *large_room* as lr, *meeting* as mt, *available* as av, *non_available* as n_av, *free* as fr and *busy* as bsy.

[3] We denote $R = sr, L = [X, Y], X = a, Y = b$ as θ_{ab}, $R = sr, L = [X, Y], X = b, Y = c$ as θ_{bc}, $R = sr, L = [X, Y], X = c, Y = a$ as θ_{ca}, and $R = lr, L = [a, b, c]$ as θ_{abc}.

[4] We denote $\langle(\leftarrow D \in \{1,2,3\}, \theta_{bc} \| n\_av(a, D), av(b, D), av(c, D)), \emptyset\rangle$ as P_{bc}, $\langle(\leftarrow D \in \{1,2,3\}, \theta_{ca} \| av(a, D), n\_av(b, D), av(c, D)), \emptyset\rangle$ as P_{ca} and $\langle(\leftarrow D \in \{1,2,3\}, \theta_{abc} \| plan(lr, [a, b, c], D)), \emptyset\rangle$ as P_{abc}.

[5] We abbreviate $\langle fr(D)@a, (\leftarrow D \in \{3\}, \theta_{ab} \| av(b, D), n\_av(c, D)), \emptyset\rangle$ as SP_1.

[6] We abbreviate $\langle fr(D)@b, (\leftarrow D \in \{2\}, \theta_{ab} \| n\_av(c, D)), \{fr(D)@a\}\rangle$ as SP_2.

[7] We abbreviate $\langle bsy(D)@c, (\leftarrow D \in \{1\}, \theta_{ab} \| \emptyset), \{fr(D)@a, fr(D)@b\}\rangle$ as SP_3.

[8] We abbreviate $\langle bsy(D)@a, (\leftarrow D \in \{1,2\}, \theta_{bc} \| av(b, D), av(c, D)), \emptyset\rangle$ as SP_4.

14. Since $fr(D)@b$ has been asked already, we do not send a question to b.
    Since $(fr(D)@b \leftarrow D \in \{1,3\}\|) \in \Delta$,
    Active: $\langle(\leftarrow D \in \{3\}, \theta_{bc}\|av(c,D)), \{bsy(D)@a, fr(D)@b\}\rangle$, P_{ca}, P_{abc}
    Suspended: SP_1,SP_2,SP_3,SP_4
15. Active: $\langle(\leftarrow D \in \{3\}, \theta_{bc}\|fr(D)@c), \{bsy(D)@a, fr(D)@b\}\rangle$, P_{ca}, P_{abc}
    Suspended: SP_1,SP_2,SP_3,SP_4
16. Suppose that $fr(D)@b \leftarrow D \in \{3\}\|$ is returned from b.
    Active: $\langle(\leftarrow D \in \{3\}, \theta_{bc}\|fr(D)@c), \{bsy(D)@a, fr(D)@b\}\rangle$, P_{ca}, P_{abc}
    Suspended: SP_1,SP_4
17. Suppose that $bsy(D)@a \leftarrow D \in \{2,3\}\|$ is returned from a.
    Active: $\langle(\leftarrow D \in \{3\}, \theta_{bc}\|fr(D)@c), \{bsy(D)@a, fr(D)@b\}\rangle$,
    $\langle(\leftarrow D \in \{2\}, \theta_{bc}\|av(b,D), n\_av(c,D)), \emptyset\rangle$, P_{ca}, P_{abc}
    Suspended: SP_1
18. $fr(D)$ is asked of c, and since $(fr(D)@c \leftarrow D \in \{3\}\|) \in \Delta$,
    Active: $\langle(\leftarrow D \in \{3\}, \theta_{bc}\|\emptyset), \{bsy(D)@a, fr(D)@b, fr(D)@c\}\rangle$,
    $\langle(\leftarrow D \in \{2\}, \theta_{bc}\|av(b,D), n\_av(c,D)), \emptyset\rangle$, P_{ca}, P_{abc}
    Suspended: SP_1
19. $(D \in \{3\}, R = sr, L = [X,Y], X = b, Y = c)$ is output as an answer constraint and $\{fr(D)@c\}$ as a set of assumed askable atoms.

At Step 6, we split a process into two processes; one active process using a default constraint and one suspended process using the negation of the default constraint. If we had to wait for an answer from the agent a, we would have to suspend this process. This is an effect of speculative computation.

At Step 16, the answer constraint for $free(D)$ is returned from b. Since this constraint entails the default constraint, nothing changes and we can continue the reduction. Therefore, in this case we receive a benefit of speculative computation.

At Step 17, the answer constraint for $busy(D)$ is returned from a. Since this constraint does not entail the default constraint nor contradicts the default constraint, we not only continue a process using the default constraint, but also resume a process using the negation of the default constraint. This is the difference between our previous work and the mechanism proposed in this paper.

6 Related Research

In computer science, there are several studies on speculative computation such as optimistic transaction in databases, three-phase transition in fault-tolerant systems, efficient execution mechanisms for functional programming and parallel logic programming. We were inspired by some of these studies, and our work is regarded as an application of the above techniques to multi-agent systems.

There is research on constraint processing in concurrent environments [Saraswat93] and distributed environments [Yokoo98].

Most related research would be constraint programming languages such as AKL (Andorra Kernel Language) [Janson91] and Oz [Smolka95] which perform a kind of speculative computation. AKL allows local speculative variable bindings in a guard of each clause until one of guards succeeds and Oz can control

multiple computation spaces each of which represents an alternative path of constraint processing. As far as we understand, however, speculative computation used in these languages is mainly motivated for or-parallel computing where multiple paths of computation are executed in parallel until one of the paths succeeds eventually. On the other hand, we regard a speculative computation as a default computation where most plausible paths of computation are executed. Moreover, they do not consider the usage of speculative computation for incomplete communication environments. However, we believe that Oz and AKL could be good platforms for implementation of speculative computation using defaults.

7 Conclusions

The following is the contribution of this paper.

- We presented speculative computation with constraint processing in multi-agent systems.
- We proposed a correct implementation of speculative computation with constraints in master-slave multi-agent systems.

The following issues remain for future research.

- Extensions to other kinds of multi-agent systems such as a system where every agent can perform speculative computation.
- Extensions in order to handle negation as failure.
- Decision theoretic analysis on effects of speculative computation.

Acknowledgments. This research is partly supported by Grant-in-Aid (No.11358004) from MEXT. We thank Jun Adachi for inviting Philippe Codognet to NII and making the collaboration among the authors possible.

References

[Janson91] Janson, S., and Haridi, S., "Programming Paradigms of the Andorra Kernel Language", *Proc. of ISLP'91*, pp. 167–186 (1991).

[Saraswat93] Saraswat, V. A., Concurrent constraint programming, Doctoral Dissertation Award and Logic Programming Series, MIT press (1993).

[Satoh00] Satoh, K., Inoue, K., Iwanuma, K., and Sakama, C., "Speculative Computation by Abduction under Incomplete Communication Environments", *Proc. of ICMAS2000*, pp. 263–270 (2000).

[Smolka95] Smolka,G., "The Oz Programming Model", Jan van Leeuwen, (Ed.), Computer Science Today: Recent Trends and Developments, *LNCS 1000*, pp. 324–343, Springer-Verlag (1995).

[Yokoo98] Yokoo, M., Durfee, E. H., Ishida, T., Kuwabara, K., "The Distributed Constraint Satisfaction Problem: Formalization and Algorithms", TKDE 10(5), pp. 673–685 (1998).

Coordinated Collaboration of Multiagent Systems Based on Genetic Algorithms*

Keon Myung Lee[1] and Jee-Hyong Lee[2]

[1] School of Electric and Computer Engineering, Chungbuk National University, and Advanced Information Technology Research Center(AITrc), Korea
kmlee@cbucc.chungbuk.ac.kr
[2] School of Information and Communication Engineering, SungKyunKwan University, Korea
jhlee@ece.skku.ac.kr

Abstract. This paper is concerned with coordinated collaboration of multiagent systems in which there exist multiple agents which have their own set of skills to perform some tasks, multiple external resources which can be either used exclusively by an agent or shared by the specified number of agents at a time, and a set of tasks which consist of a collection of subtasks each of which can be carried out by an agent. Even though a subtask can be carried out by several agents, its processing cost may be different depending on which agent performs it. To process tasks, some coordination work is required such as allocating their constituent subtasks among competent agents and scheduling the allocated subtasks to determine their processing order at each agent. This paper proposes a genetic algorithm-based method to coordinate the agents to process tasks in the considered multiagent environments. It also presents some experiment results for the proposed method and discusses the pros and cons of the proposed method.

1 Introduction

In some distributed information systems, information processing tasks are carried out by a sequence of distributed servers' operations. To perform operations, the servers may require external resources such as files, communication bandwidth, memories, and so on. In industrial manufacturing environments like flexible manufacturing systems, there are multiple workstations to perform some tasks and multiple external resources available to the workstations. Industrial products are manufactured through a sequence of workstations' processes along with usage of some resources[BLA]. This kind of distributed information processing systems and industrial manufacturing environments can be viewed as a class of multiagent systems which can be characterized as follows: In the multiagent systems, there are multiple agents, corresponding to workstations or servers,

* This work has been supported by Korea Science and Engineering Foundation through AITrc.

J. Lee and M. Barley (Eds.): PRIMA 2003, LNAI 2891, pp. 145–157, 2003.

which have their own skills to process some (sub)tasks and to require some external resources. An agent can process a subtask at a time, and a subtask can be carried out by an agent without help of other agents. External resources (e.g., files, communication bandwidth, memories, peripherals, tools, raw material, space, etc.) are either used exclusively by an agent or shared by some specified number of agents at the same time. A task consists of several subtasks in which subtasks are partially ordered, that is, the subtasks can be processed in any order as long as the order is a topological sort of the partial order imposed on them. In other words, tasks have predetermined plans about how to process them in the form of a collection of partially ordered subtasks.

We are interested in coordinating agents to perform the tasks in an efficient manner in the above-mentioned multiagent environment. The coordination work can be divided into two subproblems: task allocation to distribute subtasks among agents, and task scheduling to determine an execution order of allocated subtasks at each agent. These two subproblems are known to be NP-complete, which means it is believed that there is no way to get an optimal solution in polynomial time.

This paper proposes a genetic algorithm-based coordination method for agent collaboration in the multiagent environment. The method is a centralized coordination method in which there is a broker agent which takes charge of allocating and scheduling subtasks for the entire system. The broker agent coordinates agents according to the results obtained from a two-layered genetic algorithm of which upper layer plays the role of searching for an efficient task allocation and of which lower layer finds an efficient task schedule corresponding to the task allocation proposed by the upper layer.

The paper is organized as follows: Section 2 briefly reviews the coordination approaches in multiagent system literature. Section 3 presents the proposed genetic algorithm-based method to coordination of multiagent systems. Section 4 shows some experiment results of the proposed method and finally Section 5 draws conclusions.

2 Coordinated Collaboration

Task allocation is a crucial part of multiagent coordination which distributes tasks among all the agents concerned. The task allocation methods have been actively studied so far and they can be classified into two modes: centralized allocation and distributed allocation[FER,WEI].

In a centralized mode, two cases may arise: One is the case that a multiagent system has a hierarchical subordination organization where the superior agent will order subordinates to carry out tasks[FER]. The agent organization itself implies how to allocate tasks to agents, and thus the flexibility is very poor. The other is a method to have a special agent called broker or trader which manages all the allocation procedure by centralizing the requests from the clients and the bids for service from the servers, to match up the agents.

In a distributed allocation mode, each individual agent tries to find a service it needs from servers. We can find two approaches in this mode: allocation by acquaintance network and allocation by contract net[FER,WEI]. In systems of acquaintance network allocation, it is assumed that each agent has a skill table for the agents it knows. When an agent has a task carried out by other agents, it applies in turn to each of the agents it know which have the desired skill until one of them accepts. The contract net allocation makes use of the protocol for the drawing up of contracts in public markets in order to allocate tasks. An agent that needs a service of other agents becomes a bidding manager and the other agents able to provide the service become bidders in a market. To allocate a task, the contract net protocol goes through four stages: In the first stage, the manager sends a description of the task to perform to all those it considers able to respond or to all agents of the system. In the second stage, the bidders draw up proposals which they submit to the manager. The manager receives and evaluates proposal and awards the contract to the best bidder in the third stage. Finally, the bidder which has been awarded the contract and which therefore becomes the contractor sends a message to the manager, to signify that it is committing itself to do it. The acquaintance network allocation and the contract net allocation approaches decide which agent to carry out the task by taking into account the current status about workload, availability of resources, and so on. Compared to the centralized allocation, these distribution allocation methods have difficulty in taking into account the overall performance of the entire system. Once subtasks of tasks are assigned among agents, the agents have to decide the processing order of the assigned subtasks in a way to maximize the performance of the entire system. Due to the restrictions on resources, deliberate task scheduling is sometimes necessary especially when resources are insufficient and agents are competing for them.

In the considered multiagent systems, the plans about how to process tasks (i.e., the sequence of subtasks and the imposed precedence relations among subtasks) are given in advance, but it is not determined yet which agent will take charge of which subtask and, in addition, the processing order of subtasks at agents. In multiagent system literature, the planning itself is one of main concerns. The planning organizations can be divided into three modes: centralized planning for multiple agents, centralized coordination for partial plans, and distributed planning[FER]. Centralized planning for multiple agents assumes that only a single planner agent exists, which makes plans and organize actions for all agents. The actions of plans can be allocated to agents either in a dynamic or static way. The dynamic allocation can be carried out by a contract net mechanism. In static allocation, the agents for actions are fixed during the planning or after planning. In centralized coordination for partial plans, each agent has its own goal or task and independently draws up its own partial plan to achieve its goal, which it sends to the coordinator. The coordinator combines all the partial plans into a single overall plan, either by arranging actions in order or by determining the necessary synchronization points so as to eliminate conflicts. In distributed coordination for partial plans, agents build independently their own

partial plans and exchange each other information relating to their plans and their goals so that each of them may achieve their goals. There have been a lot of researches on distributed coordination for partial plans[DEC,DUR,DES].

The task allocation and scheduling problems discussed are known to be NP-complete[RUS]. Therefore there are yet no approaches to guarantee an optimal solution. In our problem setting, the considered agents are assumed to take considerable time to process subtasks, i.e., they are not expected to reactively behave. Thus this setting allows us to have somewhat enough time to find an efficient task allocation and efficient schedules for agents. It is assumed that a set of tasks to be carried out are given in advance at once. Later we will discuss how the proposed method can be applied to the multiagent systems to which tasks are entered in an on-line fashion. This paper proposes a genetic algorithm-based method to coordinate the agents to process the tasks. The proposed method follows a centralized approach where there is a broker agent which maintains and monitors all information about the other agents, and performs both coordinated task allocation and scheduling using a genetic algorithm. The genetic algorithm-based approaches are advantageous in the following aspects: First, they provide a solution at any moment during their search process even though it is not always optimal. Second, solutions obtained in any way from other methods can be used in the genetic algorithms since they can be encoded into chromosomes (i.e., as candidate solutions). Third, empirically it has been recognized that genetic algorithms have been finding efficient solutions in many optimization problems at least as difficult as NP-complete problems[MIT].

3 A Genetic Algorithm Approach to Coordination of Multiagent Systems

3.1 Task Allocation and Scheduling

The coordination work of a multiagent system can be decomposed into allocating subtasks to agents and into scheduling subtasks at each agent so as to maximize the performance of the entire system. The proposed method is organized in a two-layered genetic algorithm where the upper layer genetic algorithm searches through candidate task allocations and the lower layer genetic algorithm plays the role of finding efficient schedules corresponding to the task allocation suggested by the upper layer genetic algorithm. Since multiple agents might have the skill able to process the same subtask in the considered multiagent systems, task allocation should be done so as to find most competent agent for each subtask considering workload, external resources, the throughput, and other performance factors. External resources can be either used by a single agent or shared by several agents at the same time. Since subtasks may require several resources at the same time for its processing, special care should be paid to task scheduling so as to avoid deadlock situations[DEI].

For the convenience of description, we will use the following notations:

$A = \{A_1, A_2, \ldots, A_m\}$: the set of agents, where A_i is the ith agent

$T = \{T_1, T_2, \ldots, T_n\}$: the set of tasks, where T_i is the ith task

t_{ij} : a subtask of task T_i

$PO_i = \{(t_{ij}, t_{ik}) | t_{ij}$ should be processed earlier than $t_{ik}\}$: precedence relation among subtasks of T_i

$R = \{R_1, R_2, \ldots, R_l\}$: the set of resources, where R_i is the ith resource

$C(R_i)$: the number of agents which can share R_i at the same time

$O = \{O_1, O_2, \ldots, O_l\}$: the set of all possible subtasks processed in the entire system

$PT_i(O_j)$: the time it takes for A_i to process O_j

$CA(O_i)$: the set of agents that can process O_i

$PR_i(O_j)$: the set of subtasks immediately preceding the subtask O_j in task precedence relation PO_i of T_i

$N = \sum_{i=1}^{n} n_i$: the total number of subtasks in all tasks, where n_i is the number of subtasks in T_i

Depending on the applications, the objective function in the coordination can be defined in various ways. For instance, the sum of finish times of tasks or the latest finish time of tasks can be the objective function of the coordination work. The coordination method is supposed to find task allocation and schedules to optimize the employed objective function. As the results of coordination, we will get the following scheduling information for tasks, agents, and resources: For each task T_i, its schedule is given in a set of tuples $(O_j, A_k, [t_l, t_m])$ indicating that agent A_k processes subtask O_j for the time period $[t_l, t_m]$. The schedule of an agent A_i is given in a set of tuples $(T_j, O_k, [t_l, t_m])$ indicating that agent A_i processes subtask O_k of task T_j for the period $[t_l, t_m]$. The usage schedule of resource R_i is given in a set of pairs of $(A_j, T_k, O_l, [t_m, t_n])$ telling agent A_j uses R_i to process subtask O_l of task T_k for the period $[t_m, t_n]$.

For the easy computation of objective functions, the transfer time of tasks between agents is assumed to be ignorant. Tasks are transferred to agents agents along with their schedule. Therefore, after an agent A_i carries out the assigned subtask of a task T_j, it is supposed to send the task T_j to the next agent to process T_j according to the schedule.

3.2 A Genetic Algorithm for Collaborated Coordination of a Multiagent System

The proposed method is implemented by a two-layered genetic algorithm where the upper layer genetic algorithm plays the role of finding an efficient task allocation with the help of the lower layer genetic algorithm that searches for efficient task schedules corresponding to the task allocation proposed by the upper layer genetic algorithm. The following shows the generic structures of the upper and lower genetic algorithms.

t_{11}	t_{12}	t_{13}	t_{21}	t_{22}	t_{31}	t_{32}
1	2	1	3	2	1	2

Fig. 1. A Candidate Task Allocation Encoding

procedure. *Upper-Layer-GA-for-Task-Allocation*

Step 1. Initialize the population with feasible task allocations.
Step 2. Evaluate the goodness of each candidate task allocation using *Lower-Layer-GA-for-Task-Scheduling*.
Step 3. Generate the next population using reproduction, crossover, and mutation genetic operators.
Step 4. Evaluate the goodness of each candidate task allocation in the population using *Lower-Layer-GA-for-Task-Scheduling*.
Step 5. If the termination condition holds, report both the task allocation with the best performance and its corresponding schedules for agents, tasks, and resources. Otherwise, go back to Step 3.

procedure. *Lower-Layer-GA-for-Task-Scheduling*

Step 1. Initialize the population with feasible schedules, subject to the restrictions on resources and the task allocation suggested by *Upper-Layer-GA-for-Task-Allocation*.
Step 2. Evaluate the goodness of each candidate schedule.
Step 3. Generate the next population using reproduction, crossover, and mutation genetic operators.
Step 4. Evaluate the goodness of each candidate schedule.
Step 5. If it has already gone the prespecified number of generations, return the schedule with the best performance. Otherwise, go back to Step 3.

3.3 A Upper-Layer Genetic Algorithm for Task Allocation

For the upper layer genetic algorithm, the following presents the basic components required to construct a genetic algorithm such as candidate solution encoding scheme, population initialization with feasible solutions, genetic operators to generate new generations, and candidate fitness evaluation method.

Encoding Scheme. Each candidate solution represents a way to allocate each subtask to an agent. It is encoded in an array of size N, where an array element corresponds to a subtask t_{ij}, and element value is the identifier of one of capable agents $CA(t_{ij})$. Figure 1 shows an encoded candidate task allocation, where it says that subtask t_{12} is allocated to agent A_2.

Population Initialization. Feasible solutions are the assignments of a capable agent to each subtask. The initial candidate solutions are constructed by randomly choosing one out of capable agents $CA(t_{ij})$ for each subtask t_{ij}.

Genetic Operators. The reproduction operator is a fundamental operator which copies a candidate solution to the next generation, as is. Its crossover operator produces two candidate solutions C_1, C_2 from two existing candidate solutions P_1, P_2 as follows: First, make a mask array $MASK$ of size N of which element is randomly initialized to either 0 or 1 at random. Then, to construct a new candidate C_1, copy $P_1[i]$ to $C_1[i]$ if $MASK[i] = 0$ for all i. Copy $P_2[i]$ to $C_1[i]$, otherwise. The other new candidate C_2 is constructed in a similar way.

The mutation operator produces a candidate solution C from an existing candidate solution P as follows: First, copy P to C, as is. Then, for each ith element, with mutation probability, $C[i]$ is randomly set to the identifier of one of corresponding capable agents.

Fitness Evaluation. The fitness value of a candidate allocation is that of the best schedule found by *Lower-Layer-GA-for-Task-Schedule* for the candidate task allocation.

3.4 The Lower-Layer Genetic Algorithm for Task Scheduling

The following presents the basic components of the lower-layer genetic algorithm to find efficient schedules for agents corresponding to the task allocation provided by the upper layer genetic algorithm.

Encoding Scheme. Candidate solutions are feasible schedules for the task allocation suggested by *Upper-Layer-GA-for-Task-Allocation*. They are encoded in an ordered list of subtasks, which is interpreted as a topological sort of feasible schedule where precedence constraints imposed on subtasks are kept. Figure 2 shows an encoded schedule corresponding to the task allocation of Figure 1, where (a) is the ordered list representation of a schedule and (b) is the processing orders of subtasks at each agent for the schedule encoded in (a).

Population Initialization To generate feasible candidate schedules for the initial population, the following topological sort-like procedure is used. For convenience' sake, let us use an index i from 1 to N for a subtask t_{jk}.

procedure. *Generate-a-Candidate*

Step 1. Build an empty array.
Step 2. Select randomly a number c from the range 1 to N.
Step 3. If c, corresponding to a task t_{ij} is not contained in the current array and all indices for the subtasks in $PR_i(t_{ij})$ are contained in the array, then append c to the end of the array.
Step 4. If all numbers (i.e., subtasks) from 1 to N are included in the array, return the array as a candidate solution. Otherwise, go back to Step 2.

(t_{12},A_2)	(t_{31},A_1)	(t_{11},A_1)	(t_{21},A_3)	(t_{13},A_1)	(t_{22},A_2)	(t_{32},A_2)

(a) A schedule encoding for the task allocation of Figure 1

$A_1: \ t_{31} \ t_{11} \ t_{13}$
$A_2: \ t_{12} \ t_{22} \ t_{32}$
$A_3: \ t_{21}$

(b) Processing orders of subtasks at agents for the schedule encoded in (a)

Fig. 2. Encoding and Encoded Information for a Task Schedule

Genetic Operators. To generate new candidate schedules, reproduction, crossover, and mutation operators are used. The crossover operator first chooses two ordered lists P_1, P_2 from the population by the rank-based selection method[MIT]. It selects randomly an subtask O_i from the list P_1. Then it marks all subtasks to be processed by the agent which processes O_i. It copies the marked subtasks from P_1 into a new array at the same places. After that, it copies the remaining subtasks into the new array in the order that they appear in the other parent P_2.

Its mutation operator first copies to a new list $(o_1, o_2, ..., o_N)$ a chosen ordered list from the population. Next, it selects randomly a position i from the list. After that, it finds the position index lmp of the nearest left-side subtask which is processed by the agent to process the subtask o_i. It also finds the position index rmp of the nearest right-side subtask which is processed by the same agent. Later on, it selects randomly a position p in the range from lmp+1 to $rmp - 1$. Finally, it moves the ith element o_i to the position p.

These crossover and mutation operators are proven to be preserving the imposed precedence constraints in [LEE]. These precedence preserving operators generate feasible schedules since the chromosomes are thought of as topological orders of task operations on which precedence constraints (i.e., the processing sequences) are imposed.

Fitness Evaluation. To evaluate the fitness of a candidate solution, it is needed to determine the start times and finish times of subtasks in the schedule corresponding to the candidate solution. The start time of a subtask is affected by the finish times of preceding subtasks and the availability of resources required to process the subtask.

If a resource R_j can be shared by k agents at the same time, R_j is regarded as having k *usage tracks* $R_j.Tr_1, \ldots, R_j.Tr_k$ each of which can be occupied by an agent. A resource R_j maintains an open time variable $Op(R_j.Tr_l)$ for each track $R_j.Tr_l$, indicating when the track will be available. From an ordered list, its corresponding schedule is easily obtained by listing out subtasks in agent-wise according to their appearance in the list from left to right. According to the

order they appear in the ordered list, the start time $O_i.start\text{-}time$ of a subtask O_i is determined as follows:

$O_i.start\text{-}time$ = max{ finish time of the immediately preceding subtask at the agent, max{ the earliest available times of resources required to process O_i}, max{ finish times of already scheduled subtasks belonging to the same task as O_i }}

The finish time of O_i at agent A_m is set as follows:

$O_i.finish\text{-}time = O_i.start\text{-}time + PT_m(O_i)$

Once $O_i.finish\text{-}time$ is determined, the open time $Op(R_j.Tr_k)$ of the occupied track $R_j.Tr_k$ by the agent A_m is set to $O_i.finish\text{-}time$.

The fitness value $f(L)$ of a ordered list L (i.e., candidate solution) now can be determined from the start and finish times of subtasks. These are are some examples of fitness functions which can be employed in the applications:

$f(L) = \max_{O_i \in O}\{O_i.finish\text{-}time\}$: the latest finish time of subtasks

$f(L) = \sum_{O_i \in O} O_i.finish\text{-}time$: the sum of the finish times of subtasks

$f(L) = \sum_{O_i \in O}(O_i.finish\text{-}time - O_i.start\text{-}time)$: the sum of processing time of subtasks

$f(L) = \sum_{T_i \in T} (t_{in(i)}.finish\text{-}time - t_{i1}.start\text{-}time)/|T|$: the mean flow time where $t_{in(i)}$ is the last operation of T_i

4 An Application Example

We have implemented the proposed method to examine its applicability and have applied it to several problem instances. The following shows a problem instance with 4 tasks for a multiagent environment with 5 agents and 4 external resources.

$A = \{A_1, A_2, A_3, A_4, A_5\}$
$T = \{T_1, T_2, T_3, T_4\}$
$R = \{R_1, R_2, R_3, R_4\}$
$C(R_1) = 2\ C(R_2) = 3\ C(R_3) = 1\ C(R_4) = 4$
$T_1 = \{O_2, O_4, O_6, O_7, O_1, O_5\}\ PO_1 = \{(2,6),(6,7),(2,1)\}$
$T_2 = \{O_6, O_9, O_1, O_{10}, O_8\}\ PO_2 = \{(6,1),(1,10),(9,8)\}$
$T_3 = \{O_2, O_4, O_5, O_7, O_9, O_8\}\ PO_3 = \{(4,7),(5,9)\}$
$T_4 = \{O_1, O_4, O_3, O_8, O_7, O_2\}\ PO_4 = \{(1,3),(3,2),(4,7)\}$
$CO(A_1) = \{(O_1, 2, \{R_1\}), (O_2, 2, \{R_2, R_3\}), (O_4, 4, \{R_3, R_4\}), (O_6, 3, \{R_1\}), (O_7, 4, \{R_2\}), (O_8, 5, \{R_3\}), (O_9, 6, \{R_4\}), (O_{10}, 5, \{R_3, R_4\})\}$
$CO(A_2) = \{(O_1, 2, \{R_2\}), (O_2, 3, \{R_1\}), (O_3, 4, \{R_1, R_3\}), (O_4, 4, \{R_1\}), (O_5, 6, \{R_2, R_3\}), (O_6, 5, \{R_2\}), (O_7, 4, \{R_2, R_3\})\}$
$CO(A_3) = \{(O_4, 5, \{R_1\}), (O_5, 3, \{R_2\}), (O_6, 4, \{R_3\}), (O_7, 6, \{R_4\}), (O_8, 3, \{R_1, R_2\}), (O_9, 2, \{R_3, R_4\}), (O_{10}, 6, \{R_1, R_4\})\}$
$CO(A_4) = \{(O_1, 4, \{R_4\}), (O_3, 5, \{R_3\}), (O_5, 3, \{R_1, R_2\}), (O_7, 2, \{R_1\}), (O_9, 6, \{R_2\}), (O_{10}, 5, \{R_3, R_4\})\}$
$CO(A_5) = \{O_2, 5, \{R_3, R_4\}), (O_3, 4, \{R_1, R_2\}), (O_4, 3, \{R_1, R_3, R_4\}), (O_6, 6, \{R_1\}), (O_8, 4, \{R_1\}), (O_{10}, 3, \{R_1\})\}$

We applied the implemented genetic algorithm-based coordination method to the problem instance. The next is a solution found in the experiments where the latest finish time among all the subtasks was employed as the fitness value of schedules. In the experiments, the population sizes for the upper and lower layer genetic algorithms was 60, the numbers of generations for both genetic algorithms were 30 and 25, respectively. The crossover rate for upper and lower

layer genetic algorithms are 0.7 and 0.9, respectively. The mutation operator was applied only when the crossover operator has chosen two chromosomes with the identical contents. The following task allocation and schedules accomplish the four tasks in 22 unit times.

Task Allocation and Schedules for Tasks

In the following, $(O_i, A_j, [t_k, t_l])$ indicates that subtask O_i of the corresponding task is allocated to agent A_j and scheduled to be processed during the time period $[t_k, t_l]$.

$T_1 : < (O_2, A_2, [0, 3])\ (O_4, A_3, [3, 8])\ (O_6, A_1, [9, 12])\ (O_7, A_4, [12, 14])\ (O_1,\ A_2, [14, 16])\ (O_5, A4, [19, 22]) >$
$T_2 : < (O_6, A_1, [0, 3])\ (O_9, A4, [4, 10])\ (O_1, A_2, [10, 12])\ (O_{10}, A_5, [12, 15])\ (O_8, A_5, [15, 19]) >$
$T_3 : < (O_2, A_1, [3, 5])\ (O_4, A_1, [5, 9])\ (O_5, A_3, [9, 12])\ (O_7, A_1, [12, 16])\ (O_9, A_3,\ [17, 19])\ (O_8, A_3, [19, 22]) >$
$T_4 : < (O_1, A_4, [0, 4])\ (O_4, A_2, [4, 8])\ (O_3, A_5, [8, 12])\ (O_8, A_3, [14, 17])\ (O_7,\ A_4, [17, 19])\ (O_2, A_4,\ [19, 21]) >$

Schedules for Agents

Here $(T_i, O_j, [t_k, t_l])$ indicates that the subtask O_j of task T_i is scheduled at the corresponding agent for the period $[t_k, t_l]$.

$A_1 : < (T_2, O_6, [0, 3])\ (T_3, O_2, [3, 5])\ (T_3, O_4, [5, 9])\ (T_1, O_6, [9, 12])\ (T_3, O_7, [12, 16])\ (T_4, O_2, [19, 21]) >$
$A_2 : < (T_1, O_2, [0, 3])\ (T_4, O_4, [4, 8])\ (T_2, O_1, [10, 12])\ (T_1, O_1, [14, 16]) >$
$A_3 : < (T_1, O_4, [3, 8])\ (T_3, O_5, [9, 12])\ (T_4, O_8, [14, 17])\ (T_3, O_9, [17, 19])\ (T_3, O_8, [19, 22]) >$
$A_4 : < (T_4, O_1, [0, 4])\ (T_2, O_9, [4, 10])\ (T_1, O_7, [12, 14])\ (T_4, O_7, [17, 19])\ (T_1, O_5, [19, 22]) >$
$A_5 : < (T_4, O_3, [8, 12])\ (T_2, O_{10}, [12, 15])\ (T_2, O_8, [15, 19]) >$

Schedules for Resources

Here $(A_i, T_j, O_k, [t_l, t_m])$ indicates that the resource is used by agent A_i to process subtask O_k of task T_j for the time period $[t_l, t_m]$, and $R_a.Tr_b$ denotes the track Tr_b of resource R_a.

$R_1.Tr_1 : < (A_1, T_2, O_6, [0, 3])\ (A_3, T_1, O_4, [3, 8])\ (A_5, T_4, O_3, [8, 12])\ (A_4, T_1, O_7, [12, 14])\ (A_3, T_4, O_8, [14, 17])\ (A_4, T_4, O_7, [17, 19])\ (A_4, T_1, O_5, [19, 22]) >$
$R_1.Tr_2 : < (A_2, T_1, O_2, [0, 3])\ (A_2, T_4, O_4, [4, 8])\ (A_1, T_1, O_6, [9, 12])\ (A_5, T_2, O_{10}, [12, 15])\ (A_5, T_2, O_8, [15, 19])\ (A_3, T_3, O_8, [19, 22]) >$
$R_2.Tr_1 : < (A_1, T_3, O_2, [3, 5])\ (A_3, T_3, O_5, [9, 12])\ (A_2, T_1, O_1, [14, 16])\ (A_4, T_1, O_5, [19, 22]) >$
$R_2Tr_2 : < (A_4, T_2, O_9, [4, 10])\ (A_2, T_2, O_1, [10, 12])\ (A_1, T_3, O_7, [12, 16])\ (A_3, T_3, O_8, [19, 22]) >$
$R_2.Tr_3 : < (A_5, T_4, O_3, [8, 12])\ (A_3, T_4, O_8, [14, 17])\ (A_1, T_4, O_2, [19, 21]) >$
$R_3.Tr_1 : < (A_1, T_3, O_2, [3, 5])\ (A_1, T_3, O_4, [5, 9])\ (A_3, T_3, O_9, [17, 19])\ (A_1, T_4, O_2, [19, 21]) >$
$R_4.Tr_1 : < (A_4, T_4, O_1, [0, 4]) >$
$R_4.Tr_2 : < (A_1, T_3, O_4, [5, 9]) >$
$R_4.Tr_3 : < (A_3, T_3, O_9, [17, 19]) >$
$R_4.Tr_4 : < >$

Figure 3 and 4 show the Gantt charts for the schedules for agents and schedules for resources. Figure 5 shows the performance of best and average solutions over the genetic evolution.

A trivial lower bound on the optimal makespan is the maximum of the sums of operations' processing times of tasks when each operation is performed at the fastest agent without consideration of exclusive use of resources, i.e., $max_{T_i \in T}\{\sum_{t_{ij} \in T_i}\{\min_{k \in CA(t_{ij})} PT_k(t_{ij})\}\}$. This lower bound for the above example is 16. Due to the resource constraints and competition for agents, the actual makespan must be must greater than the lower bound. Therefore, even though there are no guarantees that it found out optimal solutions, the found solution must be an efficient one.

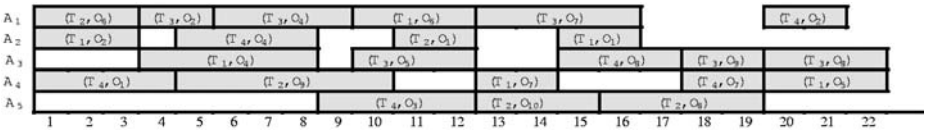


Fig. 3. Schedules for Agents

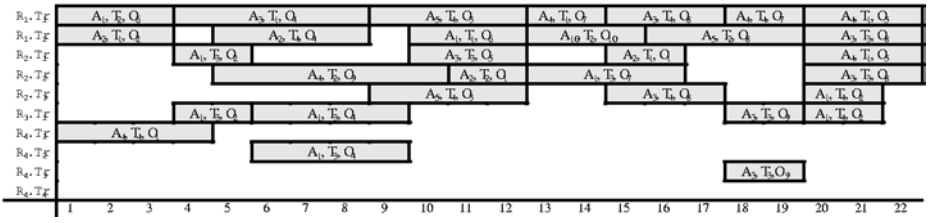


Fig. 4. Schedules for Resources

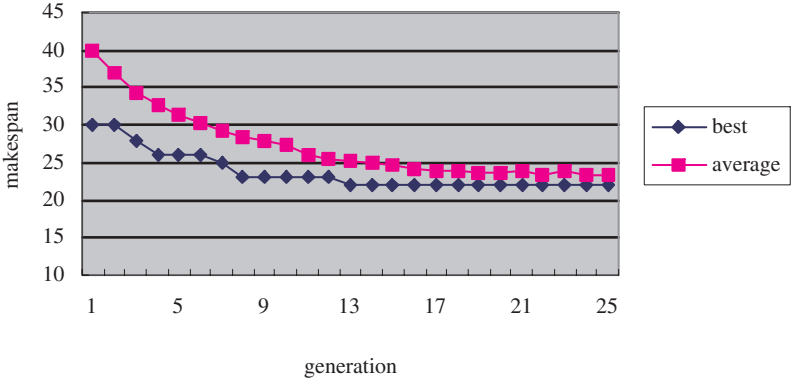


Fig. 5. Best and average performances during genetic evolution

5 Conclusions

This paper proposed a genetic algorithm-based coordination method for collaborative work in multiagent systems. In the considered multiagent systems, it is assumed at the task submission time that all tasks have predetermined plans about how to process them in the form of a collection of partially ordered subtasks. The planning is another important issue in artificial intelligence and multiagent systems. The consideration of planning aspect will bring another dimension of complexity in the problem domain. Despite of that, the proposed task allocation and scheduling approach may be employed at the afterward phase of planning when a constructed plan comes to be in action.

The concerned coordination work is to allocate subtasks among agents and to find schedules of subtasks at agents in a way to maximize the overall performance of the multiagent systems. The proposed method finds efficient task allocation and task schedules for the multiagent systems where there exist external resources which can be either exclusively used by an agent or shared by several agents, and are nonpreemptive. Even though agent may require multiple resources at the same time, the proposed method guarantees not to lead to any deadlock situations because subtasks are scheduled in the encoded order in chromosomes produced during genetic evolution and all required resources for a subtask are allocated at a time. From the experiments we have observed that the proposed method can find efficient task allocation and schedules.

It is possible for some agents to unexpectedly get failed during the execution of subtasks according to a schedule. In this case it is needed to reallocate unprocessed subtasks and reschedule them. Suppose that agent A_i got failed at time t. Usually it is not efficient to reallocate to reschedule only the subtasks of the failed agents. In addition, it is not a good idea to stop all agents and to reallocate and reschedule all the remaining unprocessed subtasks at the time failure happens. To take care of such exceptions, the notion of *scheduling window* SW is introduced during which reallocation and rescheduling are performed. At all agents but A_i, the subtasks of which scheduled processing time is overlapped partially or completely with the time interval $[t, t + SW]$ are allowed to be executed according to the current schedule. To find an efficient task allocation and task schedule for the remaining subtasks, the proposed genetic algorithm-based method can be applied. The genetic algorithm-based approaches are advantageous for failure handling because genetic algorithms maintain feasible solutions which can be applied at any moment during their searching process. This strategy can be applied to on-line scheduling where tasks arise dynamically as follows: It first sets a scheduling window at a regular interval and collects tasks to happen during the window. Then it finds efficient task allocation and schedules using the proposed method for the collected tasks during the following window.

The proposed method is a centralized approach in which a specialized broker agent takes charge of all this coordination work. There may be a communication bottleneck between the broker and the other agents due to its centralized organization. Due to inherent behaviors of genetic algorithms, the method takes some amount of time to propose efficient solutions. Therefore, this approach may be not applicable to the applications in which reactive response is of main concern, but it is suitable for such fields as manufacturing systems, information processing applications where tasks are fed in batch or tasks demand considerable processing time.

References

[FER] J. Ferber. *Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence*. Addison Wesley Longman. (1999).

[WEI] G. Weiss. *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*(eds.). The MIT Press. (1999).

[CAR] A. Cardon, T. Galinho, J.-P. Vacher. Genetic algorithms using multi-objectives in a multi-agent system. *Robotics and Autonomous Systems* 33. (2000) 179–190.

[VAZ] M. Vazquez, L. D. Whitley. A Comparison of Genetic Algorithms for the Dynamic Job Shop Scheduling Problem. *GECCO-2000*. (2000) 1011–1018.

[LAB] P. Laborie. Algorithms for propagating resource constraints in AI planning and scheduling: Existing approaches and new results. *Artificial Intelligence* 143. (2003) 151–188.

[FAN] H.-L. Fang. *Genetic Algorithms in Timetabling and Scheduling*. Ph.D. Dissertation, Univ. of Edingburgh. (1994).

[BLA] J. Blazewicz, K. Ecker, G. Schmit, J. Weglarz. *Scheduling in Computer and Manufacturing Systems*. Springer-Verlag. (1993).

[LEE] K.-M. Lee, T. Yamakawa, K.M. Lee. Genetic algorithm approaches to job shop scheduling problems: An Overview. *Int. Journal of Knowledge-based Intelligent Engineering Systems* 4(2). (2000) 72–85.

[DEI] H.M. Deitel. *Operating Systems*. Addison Wesley. (1990).

[MIT] M. Mitchell. *An Introduction to Genetic Algorithms*. The MIT Press. (1998).

[HUR] J. Hurink, B. Jurisch, M. Tole. Tabu search for the job shop scheduling problem with multi-purpose machines. In *Operations Research Specktrum*. Vol.15. (1994) 205–215.

[RUS] S. J. Russell, P. Norvig. Artificial Intelligence: A Mordern Approach. Prentice Hall. (1995).

[WIL] D. E. Wilkins, K. L. Myers. A multiagent planning architecture. In *AIPS-98*. (1998) 154–162.

[WOL] M. J. Wolverton, M. desJardins. Controling communication in distributed planning using irrelevance reasoning. In *AAAI-98*. (1998).

[KOB] S. Kobayashi, I. Ono, M. Yamamura. An Efficient Genetic Algorithm for Job Shop Scheduling Problems. In *Proc. of the 6-th Int. Conf. on Genetic Algorithms*. (1995) 506–511.

[DES] M. E. desJardins, E. H. Durfee. C. Le Ortiz,Jr., M. J. Wolverton. A Survey of Resarch in Distributed, Continual Planning. *AI Magazine.* Winter (2002).

[COR] D. D. Corkill. Hierarchical planning in a distributed environment. In IJCAI-79. (1979).

[DUR] E. H. Durfee, V. R. Lesser. Partial global planning : A coordinationframework for distributed hypothesis formation. *IEEE Trans. on Systems, Man, and Cybernetics*, KDE-1. (1991) 63–83.

[SCH] M. J. Schoppers. Universal plans for reactive robots in unpredictable environments. In *IJCAI-87*. (1987). 1039–1046.

[DEC] K. S. Decker, V. R. Lesser. Designing a family of coordination algorithms. In ICMAS-95. (1995).

[KNO] C. A. Knoblock. Planning, Executing, Sensing, and Replanning for Information Gathering. In *IJCAI-95*. (1995).

[BAR] B. J. Grosz, S. Kraus. Collaborative plans for complex group action. *Artificial Intelligence* 86(1). (1996) 269–357 .

Honesty, Trust, and Rational Communication in Multiagent Semi-competitive Environments

Ka-man Lam and Ho-fung Leung

Department of Computer Science and Engineering
The Chinese University of Hong Kong
Sha Tin, Hong Kong, China
{kmlam,lhf}@cse.cuhk.edu.hk

Abstract. This paper introduces rational communication among agents in multi-agent semi-competitive environments. In such environments, agents have both incentives to be honest and to be dishonest. Therefore, agents need to decide whether or not to trust other agents. This paper differentiates the concepts of reputation, impression and trust; proposes to include the agent's attitude towards risk in deriving the trustworthiness from the impression of another agent; and proposes to integrate an agent's impression towards the message sender, the agent's attitude towards risk, and the utility brought by believing the message in making decisions on whether to believe a message and change its action based on the message. These mimic the model in human interactions.

1 Introduction

Agent interaction has been the subject of continuous interest in multiagent system. In different environments, agents have different methods to resolve conflicts. In non-cooperative domains, Zlotkin and Rosenschein [17] use a theoretical negotiation model and a conflict resolution protocol. In cooperative domains, they also use a negotiation protocol to help agents to share their tasks [16]. On the other hand, Rosenschein and Genesereth [13] use a deal-making mechanism to enable agents to coordinate and cooperate. In contrast to the papers mentioned above, Gmytrasiewicz and Durfee [5,6,7] propose the Recursive Modeling Method (RMM) to enable agents to make decisions rationally in the absence of the pre-established protocols or conventions. In addition, Lam and Leung [8] have improved the RMM by recursive formulas, instead of using a probability approximation in the original design.

The issues about honesty and trust among agents have also been addressed by Gmytrasiewicz and Durfee [4]. However, the discussion was concentrated on cooperative environments. In this paper, we apply the Recursive Modeling Method to *semi-competitive* multi-agent environments and discuss issues related to rational communication among agents in such environments. We note that in purely cooperative environments, in which benevolent agents have a common goal of maximizing the social welfare, there is no reason for an agent not to trust its

J. Lee and M. Barley (Eds.): PRIMA 2003, LNAI 2891, pp. 158–169, 2003.

partners. On the other hand, in strictly competitive environments (such as zero-sum games), in which the self-interested agents cannot increase their utilities by forming coalitions, it is irrational for an agent to believe information provided by its competitors. However, in multi-agent semi-competitive environments, agents are in competition in some aspects, while cooperation can sometimes increase agents' utilities. Therefore, it is sometimes rational for an agent to cooperate with some other agents, while agents also have incentives to be dishonest. Therefore, agents need to decide whether or not to trust other agents when receiving information from other agents. In such a setting, the issues about honesty and trust among agents become more significant and complicated than those in purely cooperative or strictly competitive environments.

In this paper, we consider agents' decision-making processes in two different types of games: *single-round games* and *iterated games*. In a single-round game, we see the motivation to consider expected utility and degree of trustworthiness in agents' decision-making processes. In deriving the degree of trustworthiness, we propose to include the agent's attitude towards risk, in addition to reputation and impression. An iterated game is a series of single-round games, in which one round of game proceeds after another. The reputation, impression, risk-attitude, and degree of trustworthiness of agents preserve in the transition from one round to another round and the values will be updated at the end of each round. In an iterated game, we see the needs for an agent to maintain its reputation, impression, and degree of trustworthiness.

The rest of the paper is organized as follows. In the next section we briefly review the background of this work. Then in section 3, we discuss the issues related to honesty and trust among agents in a single-round game. We differentiate the concepts of reputation, impression, as well as trust. Besides, we propose to include the agent's attitude towards risk in deriving the trustworthiness from the impression of another agent. In addition, we propose that whether an agent should believe a message and follow a message, which is to change its action based on the message, is related to the agent's impression towards the message sender, the agent's attitude towards risk, and the expected utility that will bring by believing the message. Section 4 concludes the paper and discusses some possible future work.

2 Background

2.1 Impression, Reputation, and Trust: A Brief Review

There are various definitions and representations of impression, reputation, and trust. Marsh [9] is among one of the first researchers to introduce a computational model for trust. He defines *General Trust*: the amount of trusts that agent x has in agent y, which is independent of the situation, as a real number between -1 and $+1$. Marsh uses an estimation of the general trust, an agent-subjective measure about the importance of the situation, as well as utility to estimate the *Situational Trust*, which is the amount of that that agent x has in agent y for a situation. However, he has not mentioned anything about reputation.

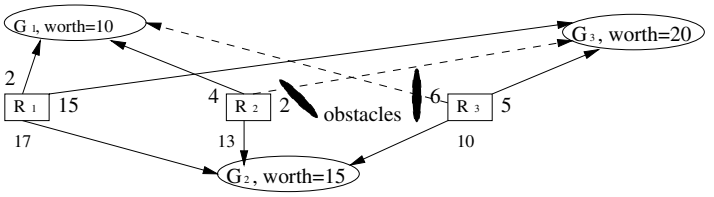


Fig. 1. Example scenario of interacting agents in a semi-competitive environment

Glass and Grosz [3] use *Brownie Points* to represent an agent's historical reputation. The value of brownie points of an agent will be increased if the agent makes a socially conscious decision, and the value will be decreased otherwise. This representation measures the opinion that a group of agents in general have about a particular agent.

Sabater and Sierra [14] proposes another reputation model. There they define *impression* that an agent has on another agent as the subjective evaluation made by an agent on a certain aspect of an outcome, and they calculate *individual-experienced reputation*, that an agent has on another agent, directly from an agent's impression database. In this model, there is a *group-experienced reputation*, that a group of agents have on a particular agent being evaluated, which is the weighted sum of the individual-experienced reputation that the member agents in the group has on the agent being evaluated. This matches the definition from the dictionary [1].

Yu and Singh [15] define reputation based on a probabilistic approach. For agent x to evaluate the trustworthiness of agent y, they calculate the reputation of agent y, which is done by combining the reputation of y as seen by a group of witness agents, as well as the reputation of the witness agents are integrated. Here, they seem to mix up trust and reputation.

Besides, Mui *et al.* [12,11] use a Bayesian approach in the computational model of trust and reputation, in which they estimate the reputation of $agent_x$ in the eye's of $agent_y$ as the probability that $agent_x$ cooperates with agent $agent_y$, that is the number of cooperation that $agent_x$ has made towards $agent_y$ out of the previous encounters. The reputation defined there is an opinion that a single agent has about a particular agent, rather than the opinion that a group of agents have about a particular agent. The *group-derived reputation* and other types of reputation are mentioned in [10]. In the computational model, Mui *et al.* define trust as the expected probability that $agent_x$ will cooperate the next time, given a history of encounters.

2.2 Recursive Modeling Method and Its Extension

Consider an example of agent interaction in a semi-competitive environment, as depicted in Fig. 1. In this environment, the agents R_1, R_2 and R_3 interact to maximize their respective payoffs. Each agent can decide to obtain one of the

goals G_1, G_2 or G_3 and the associated worths by paying a cost, except that the existence of G_3 is unknown to R_2 and the existence of G_1 is unknown to R_3. Moreover, each goal is associated with a priority ordering of agents to which the worth is given to, so that when more than one agent decides to obtain a goal, the worth of the goal will be given to the agent with the highest priority. Agents are free to communicate until all agents openly announce their choices of actions, then the game ends and the worths of the goals are given to the winning agents. Obviously there is competition among the agents. However, agents can sometimes benefit from cooperation. For example, the best action for R_1 is to obtain G_1. However, suppose R_1 has a lower priority than R_2 for G_1, but a higher priority for G_3, it is rational for R_1 to invite R_2 for cooperation by informing R_2 the presence of G_3. Consequently, the payoffs of both agents will increase.

In each game, each agent can only take one action. An agent knows which messages it received from other agents are true, and which are lies, only after the game ends. Lam and Leung [8] propose an extension to the Recursive Modeling Method (RMM) of Gmytrasiewicz and Durfee [5,6,7], which uses recursive formulas to solve an infinite belief hierarchy. This prevents having to terminate the infinite belief hierarchy explicitly by probability approximation, as in the original RMM. The above example can be solved using the the following set of recursive formulas. Due to lack of space, readers are referred to [8] the details.

$$\begin{cases} 1.\ a^{*1}_{R_1} = \arg\max_a U_{R_1}(P^{R_1-R_1}_{A_1}, a, a^{*2}_{R_1-R_2}, a^{*2}_{R_1-R_3}) \\ 2.\ a^{*2}_{R_1-R_2} = \arg\max_a U_{R_2}(P^{R_1-R_2}_{A_1}\ominus_{\{G_3\}}, a, a^{*3}_{R_2-R_1}, a^{*3}_{R_2-R_3}) \\ 3.\ a^{*3}_{R_2-R_1} = \arg\max_a U_{R_1}(P^{R_1-R_1}_{A_1}\ominus_{\{G_3\}}, a, a^{*2}_{R_1-R_2}, a^{*3}_{R_2-R_3}) \\ 4.\ a^{*3}_{R_2-R_3} = \arg\max_a U_{R_3}(P^{R_1-R_3}_{A_1}\ominus_{\{G_3\}}, a, a^{*2}_{R_1-R_2}, a^{*3}_{R_2-R_1}) \\ 5.\ a^{*2}_{R_1-R_3} = \arg\max_a U_{R_3}(P^{R_1-R_3}_{A_1}\ominus_{\{G_1\}}, a, a^{*3}_{R_3-R_1}, a^{*3}_{R_3-R_2}) \\ 6.\ a^{*3}_{R_3-R_1} = \arg\max_a U_{R_1}(P^{R_1-R_1}_{A_1}\ominus_{\{G_1\}}, a, a^{*2}_{R_1-R_3}, a^{*3}_{R_3-R_2}) \\ 7.\ a^{*3}_{R_3-R_2} = \arg\max_a U_{R_2}(P^{R_1-R_2}_{A_1}\ominus_{\{G_1\}}, a, a^{*2}_{R_1-R_3}, a^{*3}_{R_3-R_1}) \end{cases}$$

3 Decision-Making in a Semi-Competitive Environment: Single-Round Games

3.1 To trust, or Not to Trust, That Is the Question

We consider the example described in section 2.2. We note that R_1 can increase its payoff if R_2 chooses to obtain G_3. To invite R_2 for cooperation, R_1 considers it rational to send R_2 the information about G_3, which is unknown to R_2. This message, M_1, should look like this: "You can obtain the goal G_3, with worth 20, *cost*($R_1 \rightarrow G_3$) = 15, *cost*($R_2 \rightarrow G_3$) = 2, *cost*($R_3 \rightarrow G_3$) = 5 and G_3's priority list

is $< R_2, R_3, R_1 >$." However, R_3 also wants to obtain G_3, to prevent competition with R_2, R_3 considers it rational lying to R_2 and directing it to a fake goal. This message, M_2, should look like this: "You can obtain the goal G_4, with worth 24, *cost*($R_1 \rightarrow G_4$) = 50, *cost*($R_2 \rightarrow G_4$) = 4, *cost*($R_3 \rightarrow G_4$) = 50 and G_4's priority list is $< R_2, R_1, R_3 >$."

Now R_2 receives two messages: M_1 from R_1 and M_2 from R_3. If R_2 believe M_1, it will choose to obtain G_3. If R_2 believe M_2, it will choose to obtain G_4. Otherwise, it will choose to obtain G_1. The payoffs of R_2 with respect to its trust on M_1 and M_2 as well as the nature of M_1 and M_2 are summarized below:

		Nature	
		M_1 is True	M_1 is False
R_2	*Believe*	18	−2
	Not Believe	6	6

		Nature	
		M_2 is True	M_2 is False
R_2	*Believe*	20	−4
	Not Believe	6	6

Now, R_2 faces a difficult question. If R_2 makes the simple assumption that the probability for R_1 or R_3 telling the truth to be $\frac{1}{2}$, then the following expected utilities result:

	Expected Utility		Expected Utility
Believe and follow M_1	$\frac{1}{2} \times (18 - 2) = 8$	Not Believe M_1	$\frac{1}{2} \times (6 + 6) = 6$
Believe and follow M_2	$\frac{1}{2} \times (20 - 4) = 8$	Not Believe M_2	$\frac{1}{2} \times (6 + 6) = 6$

From the above table, it can be seen that both the expected utilities of believing and following M_1 as well as M_2 are higher than that of believing neither M_1 nor M_2, so R_2 will believe either M_1 or M_2, or both. However, believing M_1 will lead R_2 to obtain the goal G_3 and believing M_2 will lead R_2 to obtain the goal G_4, which are two different actions. Since each agent can only take one action, R_2 has to choose to follow either M_1 or M_2, but not both. As the expected utilities of believing and following M_1 and believing and following M_2 are the same, R_2 cannot determine whether to follow M_1 or M_2.[1]

3.2 Impression, Reputation, and Trust

Before solving the problem described above, we first differentiate the concepts of reputation, impression, as well as trustworthiness. From Cambridge Dictionaries Online [1],

- *Impression* is "the opinion you form when you meet someone or see something".
- *Reputation* is "the opinion that people in general have about someone or something, ..., based on past behaviour or character".

[1] In this paper, we assume that if an agent believes a message, the agent believes all the information provided by the message. It is arguable that an agent can, in general, choose to believe only some parts of the message. However, this will much complicate the discussion and we leave it to the future work.

– *Trustworthiness* is the property of being "able to be trusted," while *trusting* is "to have belief or confidence in the honesty, goodness, skill or safety of a person, organization or thing".

From Merriam-Webster Online [2],

– *Impression* is "a telling image impressed on the senses or the mind".
– *Reputation* is the "overall quality or character as seen or judged by people in general".
– *Trustworthiness* is the property of being "worthy of confidence".

Impression. Follow the definition in the dictionaries, we define the impression that $agent_i$ has towards $agent_j$ as:

$$imp_{ij} = f_i(\sum gain_{ij}, \sum loss_{ij}, p, n)$$

which is a real number, where $\sum gain_{ij}$ is the sum of the utility that $agent_i$ gains by believing the truths from $agent_j$, $\sum loss_{ij}$ is the sum of the utility that $agent_i$ loses by believing the lies from $agent_j$, p is the number of times that $agent_j$ tells the truth, and n is the total number of messages that $agent_i$ receives from $agent_j$. The function f_i must satisfy the following axioms:

Axiom f_{i1}: f_i is continuous.
Axiom f_{i2}: f_i strictly increases as p increases.
Axiom f_{i3}: f_i strictly increases as $\sum gain_{ij}$ increases.
Axiom f_{i4}: f_i strictly decreases as $\sum loss_{ij}$ increases.
Axiom f_{i5}: $f_i = 0$ when $n = 0$.
Axiom f_{i6}: For $\sum gain_{ij} = \sum loss_{ij}$, $f_i = 0$ when $p = n - p$, $f_i > 0$ when $p > n - p$, and $f_i < 0$ when $p < n - p$.
Axiom f_{i7}: $f_i > 0$ when $\sum gain_{ij} > \sum loss_{ij}$ and $p \geq n - p$.
Axiom f_{i8}: $f_i < 0$ when $\sum gain_{ij} < \sum loss_{ij}$ and $p \leq n - p$.
Axiom f_{i9}: $f_i < 0$ when $\sum gain_{ij} > \sum loss_{ij}$ and $p < n - p$.
Axiom f_{i10}: $f_i < 0$ when $\sum gain_{ij} < \sum loss_{ij}$ and $p > n - p$.

Axiom f_{i2} states that it is rational that impression will increase if the number of times that the message sender tell the truth to the recipient agent increases. Axiom f_{i3} means it is rational that impression will increase if the sum of the utility that the recipient agent gain by believing the messages from the sender increases. Axiom f_{i4} describes that it is also rational that impression will decrease if the sum of the utility that the recipient agent loss by believing the messages from the sender increases.

Axiom f_{i5} says that impression will be neutral if $agent_i$ receives no message from $agent_j$. For the gain in utility equals the loss in utility (axiom f_{i6}), the impression will also be neutral if the message sender has told the same number of truths and lies, the impression will be positive if the sender has told more truths than lies, and the impression will be negative if the sender has told more lies than truths.

From axiom f_{i7}, if the gain in utility is larger than the loss in utility and the sender told more truths than lies (or the same number of truths and lies), this

means that the message sender is good to the recipient agent, so the recipient agent will have a positive impression towards the message sender. On the other hand, from axiom f_{i8}, if the loss in utility is larger than the gain in utility and the sender has told more lies than truths (or the same number of truths and lies), this means that the message sender is doing harm to the recipient agent, so the recipient agent will have a negative impression.

Axiom f_{i9} is special. If the gain in utility is larger than the loss in utility, but the sender has told more lies than truths, it is very likely that the sender is performing some kinds of strategy. For example, at the first encounter, the sender tells a truth, bringing a utility of 100 to the recipient agent; but in the following nine encounters, the sender lies, which makes the recipient agent loss a utility of 90 in total, it is obvious that the sender is doing harm to the recipient agent. So, the impression in this case should be negative.

Axiom f_{i10} shows the case in which the sender has told more truths than lies, but the gain in utility is less than loss in utility, this means that the lies bring more harms to the recipient agent, so the impression is also negative.

Reputation. In an N agents environment, we define reputation of $agent_j$ as:

$$rep_j = \frac{\sum_{k=1}^{k=n} imp_{kj}}{N}$$

which is a real number. This is an average of the impression that the population has towards the agent being evaluated, which follows the definition from the dictionary. In [12,11], reputation is defined using a weighted sum of individual experience or something similar, in which they means the impression of an agent, having good reputation, towards the agent being evaluated should be counted more. However, if the agent being evaluated is only good to that particular agent, which has good reputation, but it is harmful to all other agents, then using the weighted sum to calculate the reputation will not be accurate.

Degree of Trustworthiness. In the example shown in section 3.2, it can be seen that considering expected utility alone is not enough for an agent to determine which message(s) to believe and follow when multiple messages are received. In fact, it is dangerous for an agent to believe and follow a message just because the expected utility of the message is attractive: the agent can be cheated easily. We propose that in a multi-agent semi-competitive environment, each agent should maintain an *impression*, as stated above, as well as a *degree of trustworthiness* to every message sender of the message it receives. In other words, for each ordered pair $< R_1, R_2 >$ we associate a *degree of trustworthiness of* R_2 *as seen by* R_1. We shall sometimes omit the phrase "as seen by R_1" when the meaning is unambiguous from the context.

We define the *degree of trustworthiness* to be a real number in the interval $[0, 1]$, which is to be derived from the impression that an agent has on the message sender. In human interaction, different people will have different reaction even if they were cheated by the same lie, and the amount of trust that different person

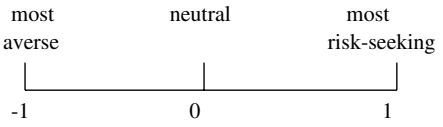


Fig. 2. Risk attitude of an agent

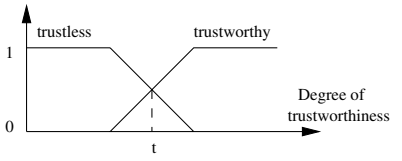


Fig. 3. Trustworthy and Trustless

has towards the liar will be different. For example, one will consider not trusting the liar anymore once he was cheated, another person may continue trusting the liar even he was cheated. This is because different person has different attitude towards risk, some do not mind taking any risk, some do not want to take any risk, while others are neutral. To model this, we propose to include the *risk attitude* of the agent in calculating trustworthiness. The risk attitude here does not mean the risk undertaken by the agent, but rather reflects the amount of risk that the agent is willing to undertake. Here, we define risk attitude, r, of an agent to be a real number in $[-1,1]$, which is shown in Fig. 2. This risk attitude is determined by the agent itself, like the personality of human, and can change over time.

Now, we define the degree of trustworthiness, t_{ij}, that $agent_i$ has towards $agent_j$ as a function of the impression that $agent_i$ has about $agent_j$, the reputation of $agent_j$ as well as the risk attitude of $agent_i$:

$$t_{ij} = f_t(imp_{ij}, rep_j, r_i)$$

The function f_t must satisfy the following axioms:

Axiom f_{t1}: f_t is continuous.
Axiom f_{t2}: f_t decreases as *imp* decreases.
Axiom f_{t3}: f_t decreases as *rep* decreases.
Axiom f_{t4}: f_t decreases as r decreases.
Axiom f_{t5}: For $imp_{i1} = imp_{i2}$, and $rep_1 \geq rep_2$,
$f_t(imp_{i1}, rep_1, r_i) \geq f_t(imp_{i2}, rep_2, r_i)$.

3.3 Making Decisions

On determining whether to believe and follow a particular message, besides considering the expected payoffs that the agent can gain by believing the message, the agent should also check the degree of trustworthiness of the message sender to see whether the message sender is *trustworthy* or *trustless*. The fuzzy linguistic terms: *trustworthy* and *trustless* are defined in Fig. 3. By using a threshold, t, and the fuzzy linguistic terms, a trustworthiness equals to 0 does not mean that the recipient agent will not trust the message sender, and a trustworthiness equals to 1 does not mean that the recipient agent will trust the message sender. Whether the recipient agent trusts the message sender depends on whether the trustworthiness of the message sender is above or below the threshold, which is determined by the recipient agent, again like the personality of human.

These two fuzzy linguistic terms help agents determine whether to believe a message or not when a message is received. If only one message is received, the agent will consider believing the message only if the message sender is *trustworthy*, and consider following the message, which is to change its action based on the message, only if the expected utility of following the message is higher than that of not following the message. On the other hand, the agent will not believe the message no matter how high is the expected utility if the message sender is *trustless*. If more than one message is received, the agent will first check each message independently to sort out a list of such messages that can be believed and followed. Then among this list of messages, the agent should choose to believe and follow the message with the highest expected utility and from the most trustworthy message sender. This implies that, if the expected utilities of all the receiving messages are the same, the agent will believe the one that is from the most trustworthy message sender. Similarly, if the degrees of trustworthiness of all the message senders are the same, the agent will believe and follow the one with the highest expected utility. On the other hand, if no such messages are sorted out, the agent will choose not to follow any message.

Now we consider the general case when an agent receives n messages from n different agents at a time. Suppose among these messages, message M_i has an expected utility ϵ_i, and the agent sending this message has a trustworthiness of t_i, where i is in $[1,\ n]$. Formally, an agent makes use of a function f_M to rank the messages and choose to follow the message that has the highest value of f_M. The function f_M has the following signature:

$$f_M : [-1, 1] \times [0, 1] \times \Re \to \Re$$

where $\Re$ denotes the real number domain.

Intuitively, the function f_M has takes the risk attitude r of the agent as the first argument, the degree of trustworthiness t_i of the message sender as the second argument and the expected utility ϵ_i of the message as the last argument, and returns a real number as the rank of the message. The function f_M must satisfy the following axioms:

Axiom f_1: f_M is continuous.

Axiom f_2: (*Adventurousness of risk-preferring agents*) There exists a value $r_0 \in \Re$ such that $f_M(r,\ t_2,\ \epsilon_1) > f_M(r,\ t_1,\ \epsilon_2)$ if and only if $r > r_0$, $t_1 > t_2$ and $\epsilon_1 > \epsilon_2$.

Axiom f_3: (*Cautiousness of risk-averse agents*) There exists a value $r'_0 \in \Re$ such that $f_M(r,\ t_2,\ \epsilon_1) < f_M(r,\ t_1,\ \epsilon_2)$ if and only if $r < r'_0$, $t_1 > t_2$ and $\epsilon_1 > \epsilon_2$.

It is obvious that the domains of the inputs of f_M are continuous, so f_M should be continuous. Besides, it is reasonable that utility will be more attractive than the trustworthiness of the message sender to a risk-preferring agent, and vice versa to a risk-averse agent. These brought about Axiom f_2 and Axiom f_3.

A recipient agent believes and follows the message with the greatest f_M value. A simple example, f_M can be defined to be the weighted sum of the expected utility of the message M_i and the trustworthiness of the message sender of M_i:

$$f_M(r,t,\epsilon) = r\epsilon_i + (1-r)t_i$$

Let us apply this function to solve the previous example. Suppose R_2 receives message M_1 from R_1 , whose trustworthiness is 0.5 and receives message M_2 from R_3 , whose trustworthiness is 0.7. Also, R_2 has a risk attitude of 0.4 and R_2 considers both R_1 and R_3 to be trustworthy. From the previous calculation, we can see that both the expected utilities of believing the messages are higher than that of not believing the messages. In addition, both the message senders are trustworthy. This means that R_2 has to determine which message to follow. The following f_M values are computed:

	$f_M(r,t,\epsilon) = r\epsilon_i + (1-r)t_i$
M_1	$0.4 \times 8 + 0.6 \times 0.5 = 3.5$
M_2	$0.4 \times 8 + 0.6 \times 0.7 = 3.62$

As M_2 gets a higher f_M value, R_2 will choose to follow M_2. From the definition of the f_M function, it is easy to see that two messages can still have the same f_M value. This means that the two messages are having the same expected utility and both are from sources with the same degree of reliability. In this case, the effect on believing and following which message will have no difference, so the agent can simply throw a dice to determine which message to believe and follow.

Theorem 1. *If two messages M_1 and M_2, with expected utilities ϵ_1, ϵ_2 and trustworthiness of message sender t_1, t_2, respectively, where $\epsilon_1 > \epsilon_2$ and $t_2 > t_1$, are sent to each one of these agents. Then there exists a constant $r_0 \in \Re$ depending only on ϵ_1, ϵ_2 and t_2, t_1, such that all the agents with risk attitude $r > r_0$ will choose to believe and follow M_1 and all the agents with risk attitude $r < r_0$ will choose to believe and follow M_2.*

Theorem 2. *Suppose there are two agents R_1 and R_2, with risk attitudes r_1 and r_2 respectively, where $r_1 > r_2$, that is agent R_1 is more risk preferring than agent R_2. Then there exist two messages M_1 and M_2, with expected utilities ϵ_1, ϵ_2 and trustworthiness of message senders t_1, t_2, respectively, where $\epsilon_1 > \epsilon_2$ and $t_2 > t_1$, such that when these two messages are sent to R_1 and R_2, R_1 will choose to believe and follow message M_1 and R_2 will choose to believe and follow message M_2.*

There is a problem with this approach, namely, a message with an extremely high utility will cause an agent to follow. First, we note that this actually mimics a real-life phenomenon occurring in human community. Second, in iterated games described in the following section, a cheated agent will decrease the degree of trustworthiness of a message sender who lied to it. Consequently, an agent will be cheated for only the first few times, and will not believe further messages from the same message sender.[2]

[2] Possibly, the agent will also become more risk-averse as a consequence.

3.4 Iterated Games

An iterated game is a series of single-round games, in which one round of game proceeds after another. The reputation, impression, and degree of trustworthiness of agents preserve in the transition from one round to another round and the values will be updated at the end of each round. So, an agent's losing credibility in one round may cause its listeners not believe it in the next round. This can become an obstacle for the agent to maximize its payoff. Therefore, it is essential for an agent to maintain its reputation, impression, as well as degree of trustworthiness, unless it is in the last round of an iterated game. Due to lack of space, We shall leave a detailed discussion to another paper.

4 Conclusion and Future Work

In a cooperative environment, there are no incentives for an agent to lie to other agents, and there are no reasons for an agent not to trust other agents. Similarly, in a totally competitive environment, there are no incentives for an agent to trust other agents. So, it is meaningless and wasteful to model whether the message listener will trust the sender or not in those environments. However, the case in Sect. 3.2 shows that in the semi-competitive environment, there are incentives for an agent to tell the truth, so there are reasons for an agent to trust other agents. Besides, there are also incentives for an agent to tell a lie, so there are also reasons for an agent not to trust other agents. The needs for an agent to determine whether to trust a message sender and to determine which message sender to trust become sensible.

In a single round game, we see the motivation to include trustworthiness, in addition to expected utility, in agents' decision-making. To do this, we first differentiate and define the terms: reputation, impression and degree of trustworthiness. Then we introduce how are these concepts integrate with the concept of risk attitude to help agent to decide on which message to believe and follow. An iterated game is a series of single-round games, in which one round of game proceeds after another. An agent should avoid losing credibility but it should maintain its reputation, impression, as well as degree of trustworthiness, unless it is in the last round of an iterated game. As future work, we are going to investigate the strategies to do so.

References

1. Cambridge dictionaries online. http://dictionary.cambridge.org/.
2. Merriam-webster online. http://www.webster.com/.
3. A. Glass and B. Grosz. Socially conscious decision-making. In *Proceedings of the Fourth International Conference on Autonomous Agents*, pages 217–224, 2000.
4. P. J. Gmytrasiewicz and E. H. Durfee. Toward a theory of honesty and trust among communicating autonomous agents. *Group Decision and Negotiation*, 2:237–258, 1993.

5. P. J. Gmytrasiewicz and E. H. Durfee. A rigorous, operational formalization of recursive modeling. In *Proceedings of the First International Conference on Multi-Agent Systems*, pages 125–132, 1995.
6. P. J. Gmytrasiewicz and E. H. Durfee. Rational coordination in multi-agent environments. *Autonomous Agents and Multi-Agent Systems*, 3(4):319–350, 2000.
7. P. J. Gmytrasiewicz and E. H. Durfee. Rational communication in multi-agent environments. *Autonomous Agents and Multi-Agent Systems*, 4:233–272, 2001.
8. K. M. Lam and H. F. Leung. An infinite belief hierarchy based on the recursive modeling method. In *Proceedings of PRIMA-2003*, 2003.
9. S. Marsh. *Formalising Trust as a Computational Concept.* PhD thesis, University of Stirling, 1994.
10. L. Mui and M. Mohtashemi. Notions of reputation in multi-agent systems: A review. In *Proceedings of Autonomous Agents and Multi-Agent Systems*, 2002.
11. L. Mui, M. Mohtashemi, C. Ang, P. Szolovits, and A. Halberstadt. Ratings in distributed systems: A bayesian approach. In *Workshop on Information Technologies and Systems*, 2001.
12. L. Mui, M. Mohtashemi, and A. Halberstadt. A computational model of trust and reputation. In *Proceedings of 35th Hawaii International Conference on System Science*, 2002.
13. J. S. Rosenschein and M. R. Genesereth. Deals among rational agents. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pages 91–99, 1985.
14. J. Sabater and C. Sierra. Regret: A reputation model for gregarious societies. In *Proceedings of Fourth International Workshop on Deception, Fraud and Trust in Agent Societies.*
15. B. Yu and M. P. Singh. Towards a probabilistic model of distributed reputation management. In *Proceedings of Fourth International Workshop on Deception, Fraud and Trust in Agent Societies*, pages 125–137, 2001.
16. G. Zlotkin and J. S. Rosenschein. Negotiation and task sharing among autonomous agents in cooperative domains. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 912–917, 1989.
17. G. Zlotkin and J. S. Rosenschein. Negotiation and conflict resolution in non-cooperative domains. In *Proceedings of the National Conference on Artificial Intelligence*, pages 100–105, 1990.

Ontology-Services to Facilitate Agents' Interoperability

Andreia Malucelli[1,2] and Eugénio da Costa Oliveira[1]

[1] LIACC-NIAD&R, Faculty of Engineering, University of Porto,
R. Dr. Roberto Frias, 4200-465 Porto, Portugal
eco@fe.up.pt
[2] PUCPR – Pontifical Catholic University of Paraná,
R. Imaculada Conceição, 1155, 80215-901 Curitiba PR, Brasil
malu@ppgia.pucpr.br

Abstract. Ontology has an important role in Multi-Agent Systems communication once it provides a vocabulary to be used in the communication between agents. It is hard to find out two agents using precisely the same vocabulary. They usually have a heterogeneous private vocabulary defined in their own private ontology. In order to provide help in the conversation among different agents, we are proposing what we call ontology-services to facilitate agents' interoperability. More specifically, in the context of the work we are doing, we intend to include these ontology-services in the framework of an Electronic Institution. Our ontology-based services will be provided through such an Electronic Institution and will be responsible for providing structural and semantic relationships between different vocabularies, useful advices on how to negotiate specific items, leading to appropriate conversations and making agreements possible.

1 Introduction

An Electronic Institution (EI) helps the management of electronic transactions based on explicit sets of institutional rules and norms. In the context of our work, the EI will help in the automatic negotiation for the formation and after the monitoring of Virtual Organizations. The electronic institution will be implemented as a framework based in agents, where the agents interact among them according to sets of restrictions mutually established.

In order to make possible the interaction between agents in a Multi-Agent Systems, it is necessary to have a communication platform, a communication language and a common ontology.

One of the big problems to communication-based agents is that each one uses different terms with the same meaning or the same term for different meanings. Once we took this problem as a challenge, representing these differences in a common ontology becomes essential. The ontology includes the entire domain's knowledge, which is made available to all the components active in an information system [6].

The use of a common ontology guarantees the consistency (an expression has the same meaning for all the agents) and the compatibility (a concept is designed, for the same expression, for any agent) of the information present in the system [10]. However, it is not sure that all the agents will use a common ontology. Usually, each

J. Lee and M. Barley (Eds.): PRIMA 2003, LNAI 2891, pp. 170–181, 2003.

agent has its heterogeneous private ontology and it cannot fully understand other agent's ontology.

Problems with heterogeneity of the data are already well known within the distributed database systems community [24] and they can be distinguished as follows:

1. Structural heterogeneity: meaning that different information systems store their data in different structures.
2. Semantic heterogeneity: considers the contents of an information item and its intended meaning. There are three main causes for semantic heterogeneity:
   i. Confounding conflicts: occur when information items seem to have the same meaning, but differ in reality, owing to different temporal contexts.
   ii. Naming conflicts: occur when naming schemes of information differs significantly.
   iii. Scaling conflicts: occur when different reference systems are used to measure the same value. An example is the use of different currencies.

Figure 1 shows two small parts of an ontology representation, with different views, but for the same domain of car assembling. Here, it is possible to see both the structural and the semantic heterogeneity types.

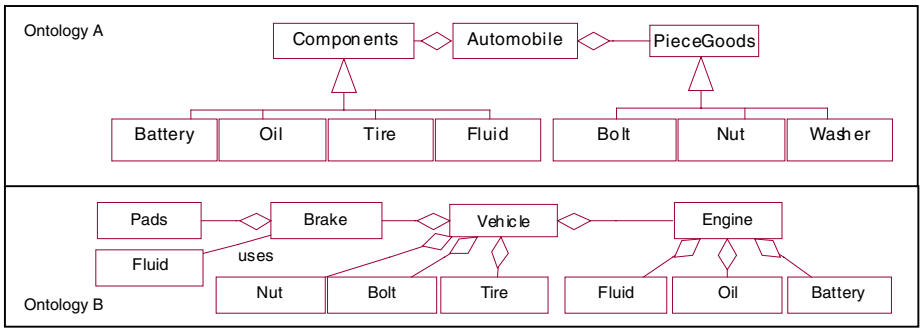


Fig. 1. Structural and Semantic Conflicts

If common domain ontology is used, it seems easier to know that people are speaking about the same subject. However, even with a common domain ontology, people may use different terms to represent the same item, the representation can be either more general, or more specific and with more details. Several difficulties are involved in providing a mapping capability between ontologies and several studies are being done to try to find a solution to that problem [24], [1].

It is also clear that agents using different ontologies make it much more difficult to establish a fruitful negotiation. An enterprise agent could provide some product with a better price and better conditions than others agents involved in a specific negotiation. Sometimes this agent does not participate in the negotiation because it does not know precisely the item (product/good/service) under negotiation; this may be caused by the fact that the enterprise agent had a different ontology in a different language or structure and could not understand the meaning of the needed item or product. Besides there problems with language and terms representation, also other problems may occur like using different currencies to denote different prices, units to represent measures or mutual dependencies of products can make the negotiation process hard.

Our ontology-based services will be responsible for providing useful advices on how to negotiate specific items, leading to appropriate conversations and making agreements possible.

Next section presents all the steps needed for the operation of the electronic institution. Section 3 discusses about ontologies and the barriers to the effective agent communication process. Section 4 presents the architecture of on proposed system. The ontology-services are explained in the section 5. An integration of services-ontology and ForEV is proposed in section 6, and finally we conclude the paper in section 7.

2 Electronic Institutions

An Electronic Institution (EI) is a framework for enabling, through a communication network, automatic transactions between parties according to sets of explicit institutional norms and rules [16]. Due to characteristics of autonomous agents, a multi-agent system can encompass an EI to facilitate agent's interoperability in the Virtual Organization formation process. Below, the generic steps where the EI has a role are presented:

i. The enterprise agents register in the EI.
ii. When some enterprise agent needs some kind of service, it informs the EI about the basic requirements.
iii. The EI send announce for the registered enterprises, which provide the service required.
iv. Each one of the enterprise agents that provides the services required send announce for the EI.
v. Those enterprises, which offer best proposes, are selected through a flexible negotiation process including learning and multi-attribute capabilities as well as distributed dependencies resolution.
vi. According to sets of rules and norms, a contract is established between partners, forming a new Virtual Enterprise.
vii. The EI monitors the partners' activities, keeping information about these activities for possible future intervention.
viii. The EI breaks up the virtual enterprise distributing the obtained profits and storage of the relevant information for future use of the EI.

In order to obtain the efficient operation of the EI, it is necessary to use a communication standard, including communication protocols, communication languages and ontologies.

In the last years, standards for agent's communication language such as KQML (Knowledge Query and Manipulation Language) and ACL (Agent Communication Language) were developed. Platforms for distributed agents communication are also available, such as JATLite (Java Agent Template Lite) and Jini. Content description language, such as KIF (Knowledge Interchange Format) and FIPA-SL (Content Language Specification) are also available.

In which ontologies are concerned, many studies are being done [2], [23], [8] but, currently, there is neither a standard ontology language nor a standard ontology knowledge representation. This lack of standardization, which hampers

communication and collaboration between agents, is known as the interoperability problem [27].

3 Ontologies and Tools

Ontologies are a popular research topic in various communities such as knowledge engineering, natural language processing, cooperative information systems, intelligent information integration, and knowledge management. The reason for ontologies being so popular is in large part due to what they promise: a shared and common understanding of some domain that can be communicated across people and computers [4].

There is not a unique definition for ontology. Several definitions [6], [25], [13] have been used and some of them are contradictory. A definition accepted in the multi-agent systems area, says that an ontology is a formal representation of concepts, characteristics and relations in each specific domain, allowing the common agreement of the people and software agents, and enabling a machine to use the knowledge of some application, multiple machines to share knowledge and still enabling the knowledge reuse [13].

Agents may use different ontologies to represent their view of a domain. Each domain can be specified in many different ways and this ontology mismatch is a question under intensive research.

[24] Presents three different directions on how to employ the ontologies: (i) single ontology approach, (ii) multiple ontology approach, (iii) hybrid ontology approach. Figure 2 illustrates the three architectures derived from these approaches:

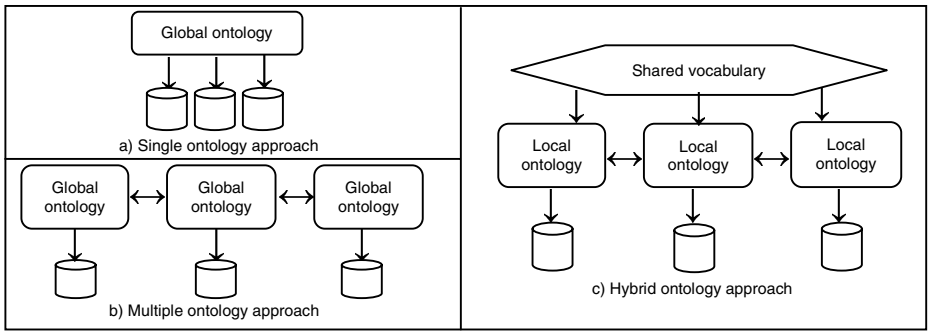


Fig. 2. The three possible ways for using ontologies

i. Single ontology approach: uses a global ontology providing a shared vocabulary for the specification of the domain semantics. All information sources are related to a global ontology. The global ontology can also be a combination of several specialized ontologies.
ii. Multiple ontology approach: each information source is described by its own ontology. In principle, the "source ontology" can be a combination of several other ontologies but it cannot be assumed that the different "source ontologies" share the same vocabulary.

iii. Hybrid ontology vocabulary: similar to multiple ontology approaches the semantics of each source is described by its own ontology. But in order to make the source ontologies comparable to each other they are built upon one global shared vocabulary. The shared vocabulary contains basic terms of the domain.

In this work, we are using the multiple ontology approach where each agent explores its own ontology. It is a decentralized and distributed approach according our multi-agent system architecture.

One would expect that by using ontology, heterogeneous agents developed by different participants and located at different places could be able to share a common ontology to provide a basis to a meaningful conversation, to ease advanced knowledge sharing between computers and human beings and to assure the accuracy as well as the quality with which agents communicate their abilities and properties to other agents.

One of the problems of using different ontologies is that there exist different representations and terminologies and there is not a formal mapping between high-level ontologies. [20] Enumerates some barriers to effective agent communication:

i. There are many different ontology languages.
ii. Different ontology languages are sometimes based on different underlying paradigms.
iii. Some ontology languages are very expressive, some are not.
iv. Some ontology languages have a formally defined semantics, some do not.
v. Some ontology languages have inference support, some do not.

Even if the exact same language is used, these ontologies can be incompatible in various ways:

vi. People or agents may use different terms for the same thing.
vii. People or agents may use the same term for the different things.
viii. A given notion or concept may be modeled using different primitives in the language.
ix. A given notion or concept may be modeled with very different fundamental underlying primitives.

A lot of different tools for developing ontologies [4] and tools for mapping, aligning [14] and reusing [15] of ontologies are available but there is no automatic way to do that, it is still a difficult task.

There already are tools for ontology translation [5] through some representation language. The uses of such tools imply offline processing and, therefore, are inappropriate for on agent-based system.

A real challenge is the automation of items translation from one ontology used by agent A to another one used by agent B.

Some research has been done trying to solve the problem of interoperability, using semantic relations, lexical taxonomy, semantic similarities, linguistic similarities of terms, taxonomic relationships, using text information.

Taxonomies are a central part of most conceptual models. Properly structured taxonomies helping bringing substantial order to elements of a model, and are particularly useful in presenting limited views of a model for human interpretation, and play a critical role in reuse and integration tasks [26].

[9] Proposes a system to compare versions of concepts. The system maintains the transformation between ontologies, some meta-data about versions, as well as the conceptual relation between concepts in different ways.

For measuring semantic similarity between words and concepts a new approach is presented by [7]. It combines the lexical taxonomy structure with corpus statistical information so that the semantic distance between nodes in the semantic space constructed by the taxonomy can be better quantified with the computational evidence derived from distributional analysis of corpus data. This is an important application for word sense disambiguation.

An approach about computing semantic similarity that relaxes the requirement of a single ontology and accounts for differences in the level of explicitness and formalization of the different ontology specifications is present by [17]. A similarity function determines similar entity classes by using a matching process over synonym sets, semantic neighbourhoods, and distinguishing features that are classified into parts, functions, and attributes.

[12] Presents two implemented methods based in linguistic similarities of terms used in the ontologies. The first looks up a dictionary or semantic network like WordNet and the second determines similarities of words based on word similarity composed from a domain-specific corpus of documents.

Other researchers are developing ontology learning trying to do a semi-automatic ontology creation using a balanced cooperative modeling paradigm [11]. Ontologies are built in different ways and with different formalisms; because of that, using text is an interesting way of eliminating the problem of different representations.

In the specific multi-agent system area some proposals have been made and implemented:

[21] Presents an upper ontology based on content reference model that provides the semantic for message content expressions. A tool is also presented which can assist agent programmers in designing message content ontologies with the Protégé.

A communication mechanism in which translators between the vocabularies of agents are generated was developed by [22]. These translators are dynamically constructed during execution stage system, and are both based on the information the agents exchange and on their underpinning ontologies. Moreover, these translators are not necessary defined for the total vocabulary of the agents, but instead, only for the parts that have been involved in communication steps.

[3] Presents the project OntoMap, whose objective is to facilitate an easy access, understanding, and reuse of general-purpose ontologies and upper-level models. They propose a language OntoMapO that provides some expressive power via single kind of expressions-binary relations between concepts. The idea is that domain specific and upper-level ontologies are loaded in the ontology of OntoMap and hosted there. This way the ontology will become accessible in all the formats supported, and the mapping will be also available in all of the supported formats.

An ontology model is presented by [19], which supports knowledge sharing in multi-agent systems where the agents are heterogeneous. The proposed model extends the usual ontology frame-based model such OKBC by explicitly representing additional information on the slot properties. This knowledge model results from a conceptual model which encompass semantic information aiming to characterize the behavior of properties in the concept description

[27] Deals with the heterogeneity by identifying an agent's internal knowledge representation with an abstract ontology representation (AOR). This AOR can be used to capture abstract models of communication related knowledge and make it possible for the agent to manipulate all elements of message in a uniform way.

An implementation of FIPA 98 Spec 12: Ontology Service is presented in [18]. There a sample application of ontology shopping service that integrates multiple database schemata using the ontology service to verify and demonstrate the specification and the implementation is presented.

4 Architecture

In an open environment where the agents can register themselves and negotiate, an ontology agent is useful to monitor all the communications and negotiations.

In our multi-agent system for VE formation there are at least 4 agents: market agent, builder agent, provider agent and ontology agent. However, it is possible and interesting for the negotiation process to have more than one provider agent.

The market agent is the entity that facilitates the meeting of agents and supports the negotiation process.

The builder, provider and ontology agents have to register themselves in the market to be able to communicate. Each agent has its own private ontology built in a private and unknown way.

Builder agent represents the enterprise interested in buying some components to build a final product. Several providers in the world may have these components with different prices and conditions. The builder agent sends a message to market to inform about the nature of components of the product (which product) that it wants to buy in order to build the corresponding product.

Provider agent represents the enterprise interested in providing some kind of product. Whenever a product is necessary, the market agent will send announce to provider agent.

Ontology agent is monitoring the whole conversation trying to help when it is necessary, providing some kind of services (discussed in section 5).

Whenever agents register, they also indicate their capabilities. This information becomes crucial for the negotiation stage. For the sake of tests, the initial scenery is the assembling car domain. We have an agent builder, which wishes to buy components to build cars of a specific model, and several provider agents, which may have, or not components to sell them.

The platform we are using to implement the multi-agent system is JATLite and the communication language is KQML. The ontology agent includes a basic domain ontology represented in XML.

The Fig. 3 shows an example of the multi-agent architecture where links represent agent's communication and the ontology agent monitoring.

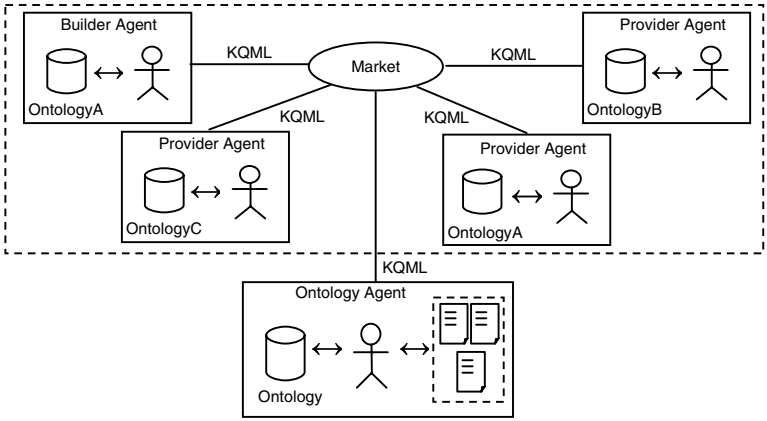


Fig. 3. Architecture of the system

5 Ontology-Services

In order to help in the communication and negotiation between agents, some ontology-services are provided:

i. Definitions or relationships between expressions.
ii. Definition of product attribute's dependencies.
iii. Capabilities for defining, modifying and deleting expressions and ontology definitions.
iv. Abilities to translating expressions between two different ontologies.
v. Possibility to learn with the ontology services already provided so that it could be possible to use this information in a future negotiation.
vi. Converting values when agents work with different metrics (like currencies).

We will also try to integrate an ontology editor to allow the definition of new ontologies.

Our proposal is that the ontology agent will monitor and help the communication process at the moment when it is happening, without having to do a mapping of all the ontologies involved.

One of the most complex services is to help in the communication when the agents use different terms for the same meaning. When a provider agent receives announce of some product/component, the provider agent has to answer if it has the component to provide or not. If the answer is "I don't know", the ontology agent understands that this agent may have the component but it may not know the meaning of the used term.

Therefore, in order to help the resolution of this incompatibility, the ontology agent:

i. Searches for synonymous in its own ontology to find equivalent terms known by the receiving agent.
ii. Searches for relations like "is", "is-a", "part-of", and "compose-by", in order to find out a different way of expressing the unknown term.

iii. Searches for characteristics of the component, which can help the provider ontology to find another component with the same characteristics.
iv. If all the previous steps are not enough, the ontology agent may not know the terms in appreciation, and then it looks in texts for new meanings and characteristics. A database of domain texts is used for the ontology agent to find terms. The text processing stage can help to find terms of a domain and can help to find relations between concepts. Using texts is not an easy work but it looks natural to use texts based on natural language to understand the communication, which is going on.

If in the end of all the process it was possible to find some matching terms, it becomes necessary to establish a protocol for confirmation when the provider agent sends the term that is the probable matching term and the ontology agent makes the validation.

Another service provided by ontology-services agent is to help during the negotiation when agents are using different currencies. The ontology agent accesses a free web service that gives the currencies in real time. This way the agent can choose the currency, which will be used during the negotiation.

The dependencies are also a problem about which the ontology services can help. Some products already have dependencies, which are better to be known during the negotiation. For example, when the agent is buying a light for the car, there are several kinds of lights, brake light, cross light, fog light, and others. In order to be able to insert the light in the car, it is necessary to buy a light-holder. There is already a standard, which says which light-holder is necessary for each kind of light. So, the ontology agent can help during the negotiation giving this kind of information.

6 Integration of Ontology-Services in ForEV

ForEv (acronym for Virtual Enterprises Formation equivalent in Portuguese) is an appropriate computing platform, which includes and combines negotiation's methods in the context of Multi-Agent Systems, which is suitable for Virtual Enterprise formation scenario [10].

The negotiation's methods try to satisfy the electronic market dynamics and competitiveness requirements. The negotiation process is between independent enterprises leading to the Virtual Enterprise formation, and relies on an iterative, multi-attribute negotiation protocol appropriated for the context of a business activity. In order to solve the simultaneous partial inter-dependent negotiations dependency problem, arising during the Virtual Enterprise formation phase, a specific algorithm was developed. This algorithm is a decentralized distributed dependency satisfaction problem solver, which is based on the progressive utility decrement concept. Through the application of this algorithm, the Multi-Agent System is able to, in a non-centralized way, select that solution which minimizes the total agent's utility decrement value for those agents involved in that specific dependency.

Agents represent the enterprises and customers in the system. There is a Market Agent (MA) and several Enterprise Agents (EA). The MA and EAs all together

through a website cooperating with the objective of providing or getting goods, keep their own preferences and objectives.

The goods are described in an ontology, which is known and understood for all the participants. The system has also a register agent responsible for building and maintenance of the domain ontology. This agent is responsible for distribute the necessary information about this ontology when asked for some agent. The ontology-services can be integrated in ForEv allowing the use of heterogeneity ontology.

Integrating the ontology-services with ForEV makes ForEV more opened, enabling establishment of the negotiation process between agents with different ontologies although representing the same domain of knowledge.

7 Conclusions

A big challenge for communicating agents is to resolve the problem of interoperability. Through the use of a common ontology it is possible to have a consistent and compatible communication.

Several problems involved in the resolution of interoperability are difficult to be solved, at least nowadays, but it is important to look for possible ways to resolve parts of the problem.

This paper was focused on the use of multiple ontology approach, where each agent explores its own ontology. It is a decentralized and distributed approach according to our multi-agent system architecture.

Ontology-based services were proposed to help the communication and negotiation between agents. In an open environment where agents can register themselves and negotiate, ontology agent is used to monitor and lead the communication process at the moment when it is happening, without having to do a mapping of all the ontologies involved.

The integration of ontology-based services with ForEV is also proposed to make this computing platform more opened and to allow ForEV to use the heterogeneity ontology by the agents during negotiation process.

References

1. Corcho, O., Gómes-Pérez, A.: Solving integration problems of e-commerce standards and initiatives through ontological mappings. In Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI-01), Workshop: Ontologies and Information Sharing, Seattle, USA (2001)
2. Fensel, D., Horrocks, I., van Harmelen, F., Decker, S., Erdmann, M., Klein, M.: OIL in a nutshell. In: Knowledge Acquisition, Modeling, and Management, Proceedings of the European Knowledge Acquisition Conference (EKAW-2000), R. Dieng et al. (eds.), Lecture Notes in Artificial Intelligence, Springer-Verlag. (2000)
3. Dimitrov, M., Kiryakov, A., Simov, K. I.: OntoMap: upper-Ontology Service Agent. In Proceedings of the Workshop on Ontologies in Agent Systems. 5th International Conference on Autonomous Agents. Montreal, Canada. (2001)

4. Duineveld, A. J., Stoter, R., Weiden, M. R., Kenepa, B., Benjamins, V. R.: WonderTools? A comparative study of ontological engineering tools. In Proceedings of the 12th Banff Knowledge Acquisition for Knowledge-Based Systems Workshop. (1999)
5. Gruber, T. R.: A Translation Approach to Portable Ontology Specifications. Technical Report KSL 92–71. Knowledge Systems Laboratory. (1993)
6. Huhns, M. N., Singh, M. P.: Readings in Agents. Morgan Kaufmann Publishers, INC. San Francisco, California (1997)
7. Jiang, J. J., Conrath, D. W.: Semantic Similarity Based on Corpus Statistics and Lexical Taxonomy. In Proceedings of International Conference Research on Computational Linguistics. Taiwan (1997)
8. Karp, P.D., Chaudhri, V. K., Thomere, J.: XOL: An XML-based ontology exchange language. Version 0.4 http://www.ai.sri.com/pkarp/xol/xol.html. (1999)
9. Klein, M., Kiryakov, A., Ognyanoff D., Fensel, D.: Finding and specifying relations between ontology versions. In Proceedings of the ECAI-02. Workshop on Ontologies and Semantic Interoperability. Lyon. (2002)
10. Macedo, A. P.: Metodologias de Negociação em Sistemas Multi-Agentes para Empresas Virtuais. Tese de Doutoramento, Faculdade de Engenharia, Universidade do Porto (2000)
11. Maedche, A., Staab, S.: Ontology Learning for the Semantic Web. IEEE Intelligent Systems, 16(2), (2001)
12. Mitra, P., Wiederhold, G.: Resolving Terminological Heterogeneity in Ontologies. In Proceedings of the ECAI-02. Workshop on Ontologies and Semantic Interoperability. Lyon. (2002)
13. Noy, N. F., McGuinness, D. L.: Ontology Development 101: A Guide to Creating your First Ontology, Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880, (2001)
14. Noy, N. F., Musen, M. A.: Evaluationg Ontology-Mapping tools: requirements and Experience. Workshop on Evaluation of Ontology Tools at EKAW'02 (2002)
15. Pinto, H. S., Martins, J. P.: Ontology Integration: How to perform the Process. Proceedings of the IJCAI-01 – Workshop on Ontologies and Information Sharing, Seattle, USA. (2001)
16. Rocha, A. P., Oliveira E.: Electronic Institutions as a framework for Agents' Negotiation and mutual Commitment. 10th Portuguese Conference on Artificial Intelligence, Proceeding of EPIA2001, LNAI, Springer Verlag, Portugal (2001)
17. Rodríguez, M.A., Egenhofer, M. J.: Determining Semantic Similarity Among Entity Classes from Different Ontologies. IEEE Transactions on Knowledge and Data Engineering 15(2) (2003) 442–456
18. Suguri, H., Kodama, E., Miyazaki, M., Nunokawa, H., Noguchi, S.: Implementation of FIPA Ontology Service. In Proceedings of the Workshop on Ontologies in Agent Systems. 5th International Conference on Autonomous Agents. Montreal. Canada. (2001)
19. Tamma, V., Bench-Capon, T.: A conceptual model to facilitate knowledge sharing in multi-agent system. In Proceedings of the Workshop on Ontologies in Agent Systems. 5th International Conference on Autonomous Agents. Montreal, Canada. (2001)
20. Uschold, M.: Barriers to Effective Agent Communication. Position Statement. Workshop on Ontologies in Agent Systems. Montreal, Canada (2001)
21. van Aart, C., Pels, R., Caire, G. Bergenti, F.: Creating and Using Ontologies in Agent Communication. In Proceedings of 1st International Conference on Autonomous Agents & Multiagents (AAMAS´02), Second International Workshop on Ontologies in Agent Systems (OAS2002). July. Bologna. Italy (2002)
22. van Eijk, R. M., Boer, F. S., van der Hoek, W., Meyer, J-J Ch.: On Dynamically Generated Ontology Translators in Agent Communication. International Journal of Intelligent Systems, vol. 16. (2001) 587–607
23. van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D., Patel-Schneider, P. F., Stein, L. A.: OWL Web Ontology Language Reference: W3C Working Draft 31 March 2003. http://www.w3.org/TR/2003/WS-owl-ref-20030331. (2003)

24. Wache, H., Vögele, T., Visser, U., Stuckenschmidt, H., Schuster, G., Neumann, H., Hübner, S.: Ontology-Based Integration of Information-A Survey of Existing Approaches. In Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI-01), Workshop: Ontologies and Information Sharing, Seattle, USA (2001)
25. Weiss, G.: Multiagent Systems. A Modern Approach to Distributed Artificial Intelligence. The MIT Press Cambridge. Massachusetts. London. England (1999)
26. Welty, C., Guarino, N.: Supporting ontological analysis of taxonomic relationships. Data&Knowledge Engineering 39 (2001) 51–74
27. Willmott, S., Constantinescu. I, Calisti, M.: Multilingual Agents: Ontologies, Languages and Abstractions. In Proceedings of the Workshop on Ontologies in Agent Systems. 5th International Conference on Autonomous Agents. Montreal. Canada. (2001)

An Ontology-Based Intelligent Agent System for Semantic Search in Medicine

Jung-Jin Yang

School of Computer Science and Information Engineering
The Catholic University of Korea
YuckGok 2-Dong San 43-1WonMi-Gu BuCheon-Si KyungGi-Do, Korea
Tel: +82-32-340-3377, Fax: +82-32-340-3777
jungjin@catholic.ac.kr

Abstract. The volume of literature in the medical domain is expanded exponentially and generating a proper query for finding related information puts a cognitive burden on users, due to a full of professional keywords in the domain. We describe an ontology-based information retrieval agent system in medicine through bio-related literature database MEDLINE, in particular. The task of interface agent system here is to proactively help user to reformulate queries in order to get useful and relevant information by utilizing both existing medical ontologies and its own agent ontology. Agent ontology is in the form of the Semantic Web languages. The goal of the research is to improve the quality of information retrieval and reduce user's cognitive load during information search by employing agent system for semantic search. Empirical results are evaluated and discussed for agent performance.

Keywords: Information Retrieval, User Interface Agent System, Semantic Web, Semantic Search

1 Introduction

Information retrieval is an everyday activity. It can be a time intensive and cognitively demanding task due to the information overload and its complexity through the advent of the Internet. The difficulty of retrieving relevant information increases further in the professional domains such as medicine. Formulating adequate query itself imposes a severely heavy cognitive burden on users, due to the utilization of a full of professional keywords. While the need to help users reduce their work and to improve search results has been emerged, methods for systematic retrieval and adequate exchange of relevant information are still in their infancy.

The recent report of completing human genome project indicates a potential shift of preventing and treating paradigm of disease. It is known to provide the foundation of leaping current medicine from experience-based and information-based ones toward prediction-based medicine. However, achieving the beneficiary goal still needs a way to retrieve and analyze the exponentially expanding bio-related information for finding correlation among information with various and sensitive factors. Moreover, research on bio-related fields naturally applies knowledge discovered to the current problem and make inferences to extract new information. It

J. Lee and M. Barley (Eds.): PRIMA 2003, LNAI 2891, pp. 182–193, 2003.

is necessary for researchers to communicate and exchange knowledge in order to extract further knowledge based on shared concepts. The shared concepts and data containing information need to be defined and used in a coherent way. Ontologies that specify terms and relationships among terms [1][2][3] facilitate sharing of data and knowledge among computational biologists. Also expanded volume of bio-related literature in MEDLINE needs a systematic search strategy and reviews, for the literature contains inter-related information and newly discovered knowledge in it. However, it is published in an electronic form that is not accessible by machines and that makes it hard to gain correlated information from it.

Overall, agent paradigm is a best fitted technology for finding correlated information by weaving through scattered data by applying information in hand and enhancing existing knowledge from newly learned information in dealing with the problems mentioned above. We employ the Semantic Web technique for augmenting targeted data with markup that describes some meaning of the data and encodes it in a form that is suitable for machine understanding. The Semantic Web community addresses these issues by defining standard mark-up languages like RDF, RDF Schema, DAML+OIL, and OWL[4]. These languages provide features to represent both shared concepts onto an ontology and data in a form to be processed.

We have built an ontology-based information retrieval agent system in medicine through bio-related literature database MEDLINE, in particular. The goal of this research is both to improve the quality of information retrieval and to reduce user's cognitive load during information search. This paper describes our preliminary design and implementation of such a system called OnSSA (an Ontology-based Semantic Search Agent) that automates systematic retrieval of literature in medicine by utilizing the Semantic Web languages. The rest of the paper is organized as follows: the next section discusses the problem in our work. Then we describe the framework, information retrieval process, empirical results, related work and future work in order.

2 Problem

Most users make a query on general-purpose search engine without having professional knowledge about what they are looking for. It is rather an opportunistic search instead. In order to gain effective information retrieval, users check on results of previous query returned from search engine and reformulate a query either a bit too general or specific. For example, suppose a general user or an expert puts a query with 'deafness' to PubMed interface, a biomedical search engine as in Fig. 1. The system returns 33033 numbers of documents as results retrieved from MEDLINE database. Unlike general-purpose search engines, professional engines of retrieving professional literature references are often hard to formulate queries without having proper knowledge of the domain.

To resolve this problem, our agent system utilize agent ontology to generate a *relevant* query of a keyword given by a user utilizing existing medical ontologies as well. MEDLINE is an on-line bibliographic database created by the U.S. National published since 1965. Multiple interfaces are available for searching MEDLINE; each differs in appearance and internal logic but seeks the same target content. While these interfaces are valuable to some degree, unfortunately, the few published strategies for identifying these articles involve MEDLINE interfaces not widely available outside of

academic medicine [5][6][7]. Therefore, we retrieve literature through PubMed that is a MEDLINE interface freely available via the Internet from the NLM. For the effectiveness test of semantic search within MEDLINE database, the results of different but relevant queries related to a query given by a user are analyzed and presented for comparison in Section 5.

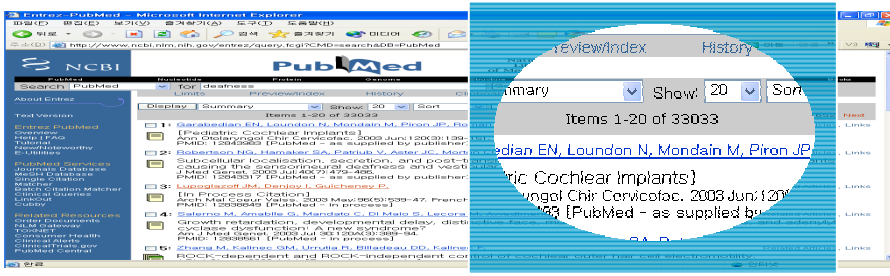


Fig. 1. An Example of a Query given by a User

3 Framework

We start this section by describing the overall process in OnSSA and then describe in detail how the agents help the system build the adapted query.

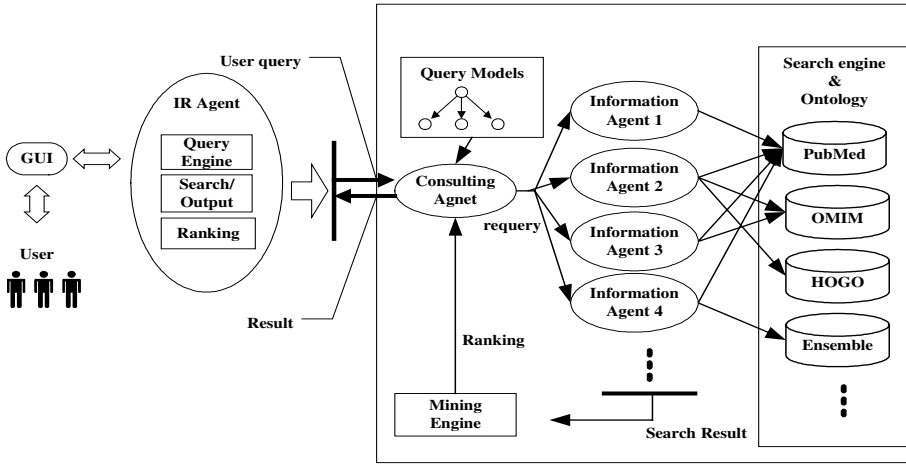


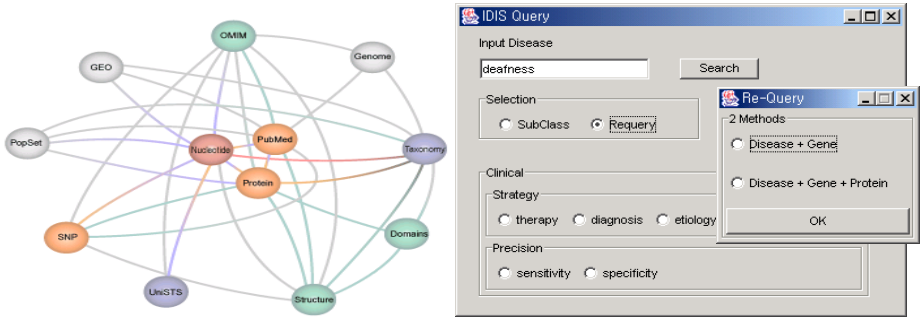
Fig. 2. System Overview

OnSSA consists of four modules as shown in Fig. 2. The *input module* proactively constructs the queries on behalf of the user as it utilizes existing medical ontologies and query models of the system. For example, if a user queries with a disease name, then the interface agent lets a query be a disease name with relevant gene names. It is then confirmed with a user to make sure the query regenerated is what the user wants to search. A consulting agent is responsible for interacting with the user. It refers to

the query model to generate a relevant query, gets a confirmation on the query reformulated from the user, finds an agent list for matching the query in hand and assigns the query to distributed information agents, and recommends selected information to the user.

The ontology built on these Semantic Web standard languages could include terminologies of this field – class taxonomy in RDF Schema, properties defined for the classes can model a schema of an agent's mental state, the state itself is represented by an RDF graph, it comprises facts defined a priori as well as knowledge acquired through the perception system. The information agents take different semantic search strategies for information retrieval.

User interface accepts the user's query and transfers it to the IRagent system in Fig. 2. The IRagent system contains three modules: *Query engine* module to reformulate a query given, *Search/output* module to orchestrate systematic search using distributed information agents, and *Ranking* module to take over and rank the results retrieved by the information agents. User's query is sent to the Query managing agent. It is transferred to the Prolog-like clause in the first order logic (FOL) form. Query managing agent uses ontology network and generate relevant queries through inferences. Reformulated queries are confirmed through user's feedback and distributed to information agents. The kinds of queries given to information agents can be searches for disease, disease-gene, disease-gene-protein relations and more. Information agents use different biomedical ontology depending on either different queries or agents' search strategies.



(a) (b)

Fig 3. (a) Biomedical Ontology and Search Engines (b) GUI agent

Task or context-specific analysis of biological data requires exploiting the relations between terms used to specify the data, to extract the relevant information and to integrate the results in a coherent form. Biomedical information is rather well-defined in terms of classification and taxonomy and it already has many large volumes of medical ontologies for different but particular purposes. Gene ontology (GO) for classification of medical terminology, G2D for disease to gene, Hugo for human gene nomenclature, OMIM (Online Mendelian Inheritance in Man): a catalog of human genes and genetic disorders, GDB (Gene database), Ensembl, and LocusLink are major instances. Those biomedical search engines based on ontologies have unique ids of their own indexing but are related with diseases and relevant genes and proteins. Figures 3(a) and 3(b) represent the relation of ontologies and GUI part of

OnSSA. Since one of the strengths of the Semantic Web is the distribution of the available information among a lot of nodes. Our distributed search systems assume the existence of several processing agents and each system provides a particular way of identifying systematic search for literature reviews for a decision-making on behalf of consumers, policymakers and clinicians. Periodically, agents review their success and report general success and selected results to the consulting agent through the mining agent. The reliability of distributed information agent is determined depending on how close and/or related the search results of individual information agents is to the query. The evaluation of search results retrieved by individual information agent is done by the mining agent and handed to the consulting agent acting as a supervisor. With the evaluation result, this supervisor enhances the whole state of the best agent with the selected results, exchanges agents that are not performing well, and then communicates the enhanced state as new start state to all agents while interacting with the user. The generalization and/or specialization of query are based on the query model that represents systematic search strategy at the level of user interface. The query models are enhanced with new knowledge that it has learned from the analysis of results returned.

In this paper, we mainly focus on how differently information agents search a secondary query reformulated from a first query by a user. The details of the rest modules are not addressed in this paper.

4 Agents for Information Retrieval Process

Figure 4 shows an algorithm for reformulating a query given by a user.

```
Input: User's Query
Output: Groups of literatures retrieved

Algorithm ReformulateQuery(Query) {                          //a query given by a user
        Rules[]←LoadQueryModule(Query);        //induce rules from QueryModule
        For(i=0;i<the number of Rule in Rules;i++) {          //for all rules selected
                InfoAgent←ExploreAgent(Rule[i]);             //find agents applicable from an agent list
                ResultSet[i]←LoadInfoAgent(InfoAgent,Query); //collect results retrieved by InfoAgents
        }
        LoadMinigAgent(ResultSet[])    //Results returned by agents are evaluated through a mining agent
}
```

Fig. 4. Algorithm for OnSSA

The system supports the best match ranked output retrieval with a query. DAG(direct acyclic graph)-based query models provide plausible queries based on a query by the user. For example, if a novice user inputs a disease name, either the system can regenerate the query with relevant genes and get a confirmation with the user or the user can choose a button to reformulate the query with the relevant genes. The adapted query is then distributed to multiple information agents capable of operating the query but with a different search strategy through its own ontology. The query model here represents *what* the user wants to do, while the agent ontology network is *how* the information is retrieved.

Agent Architecture

Our agent system differs from other interface agents in that the agents reformulate a query given by a user autonomously and proactively to a more adequate and relevant query to search in a massive and professional domain. Major components of an agent can be following: taxonomy of classes is represented in RDF schema, properties of a class can be modeled as a schema to represent intelligent states of an agent. The state itself is represented into RDF graphs and the graphs consist of both facts that agents should know in advance and knowledge obtained through an agent's perception. In addition, RuleML[8][9] plays major roles with following properties: First, it allows to define integrity constraints of avoiding illegal intelligent state of an agent. Second, it can describe knowledge of agent properties or that of agent's learning process through derived rules. Third, it can define reaction rules for an agent to respond to events and/or messages.

GUI Agent

A user query is reformulated based on query models and it can be combined with *clinical query* provided in both *category*: therapy, diagnosis, etiology, and prognosis and *emphasis*: sensitivity and specificity through PubMed. Both the number and contents of documents provided by PubMed are quite different depending on keywords in a query. A query given by a user is polished and re-generated to a more adequate query in the domain. This will result in reducing a cognitive load on the user in great deal, where professional knowledge is required to formulate a proper query. The adapted secondary query is transferred to the distributed information agents to improve the quality of results retrieved.

Information Agent

Individual information agent is distinguished with a particular way of identifying systematic search for literature reviews for a decision-making.

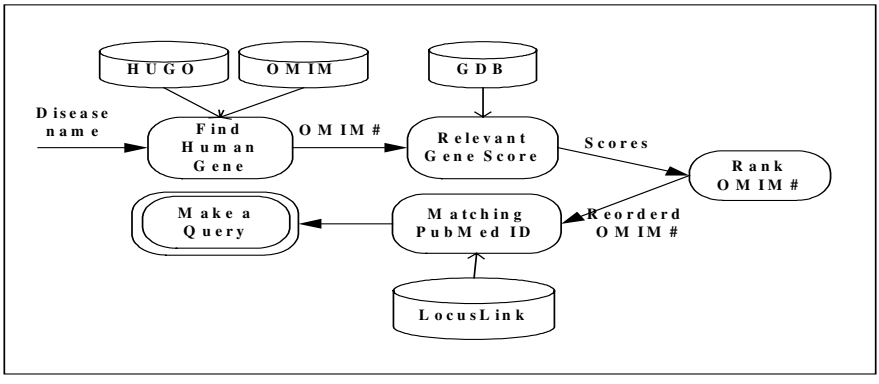
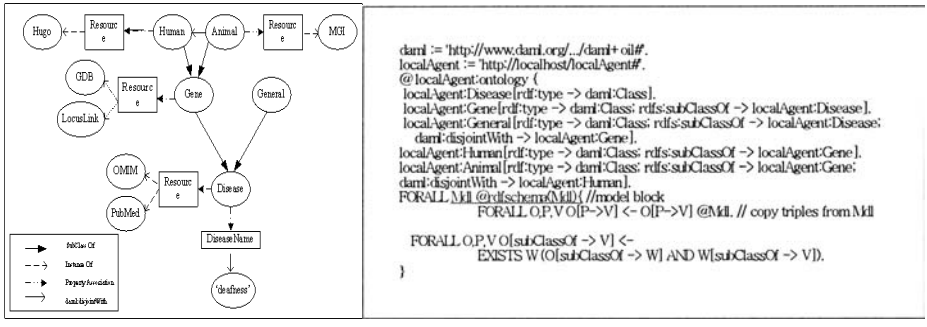


Fig. 5. The Process of Information Retrieval

Figure 5 demonstrates an example of information retrieval process taken by one information agent. The information agent utilize different medical ontologies such as Hugo, GDB, OMIM, LocusLink, in addition to its own ontology to retrieve

documents related to a query. Hugo lets the agent retrieve approved human gene names related to a disease name in the query. Each gene symbol has links to other databases. Each database provides references of each gene with its own ids and these references are correlated through gene names. GDB provides the genes' GDB ID with scores of indicating genes' relevance to the disease. Here the information agent could take different search strategies by using either OMIM references or correlation information of different references for each gene provided by LocusLink[1].

In our experiments, we examined both approaches in which both of them used GDB scores for re-ordering gene names. GDB has Hugo's gene symbols as a primary name and provides three possible accession[2] and each gene has a score ranking data[3]. The information agent generates PubMed ids matching with the genes through OMIM id. It formulates an adaptive query with both a disease name and the PubMed ids of relevant genes and submits the query to PubMed for literature retrieval. Figure 6 (a) shows an agent ontology of this information agent in a ontology network and Fig. 6 (b) presents the agent ontology into TRIPLE/DAML+OIL language. The details of TRIPLE/DAML+OIL can be found in [10] The relations among Hugo, GDB, and OMIM can be represented in DAML+OIL as in Fig. 7.



(a) (b)

Fig. 6. Agent Ontology (a) agent ontology network (b) TRIPLE/DAML+OIL

Mining Agent

Each group of retrieved literatures by an information agent is examined for relevancy to a query given. The relevancy level is a real number indicating how close the literatures are to the query. Each query term is denoted as *t* and its associated relevancy level as *L(t)*. An mining agent computes *L(t)*, after each information agent returns a group of documents, by: $L(t) = (0.5 + 0.5\ freq_{i,q} / max_i\ freq_{i,q}) * log\ n/m$ with *n* as the number of relevant documents containing this term *t* and *m* as the number of relevant documents. *L(t)* of a simple query only through PubMed is used as

[1] LocusLink has correlation information among external references and we used loc2cit data for linking OMIM number and PubMed id.

[2] 3 accession: Biological data(GDB:xxx), Citation(CIT:xxx), People or Organization (REG:xxx).

[3] GDB's keyword search is a free-text search using Oracle tools, which locates objects that contain the specified query term(s) and can rank them accorting to the frequency with which the term(s) appear. All query terms are case insensitive and allow the use of wildcards.

a baseline of others for comparison. With results periodically evaluated and reported, a consulting agent can conduct coordination or negotiation among agents while keeping interactions with the users for changes of user interests. Depending on the expertise of the user on this domain, the user can provide feedback directly. Another role of a mining agent is the learning process. It can distinguish the user interests with subject matters from retrieved information and enhance the user profile and/or agent ontology network. The validation of agent ontology is needed.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
  <rdf:RDF
        xmlns:daml="http://www.daml.org/2001/03/daml+oil#"
        xmlns:dc="http://purl.org/dc/elements/1.1/"
        xmlns:ns0="file:/C:/HUGO.daml#"                    xmlns:oiled="http://img.cs.man.ac.uk/oil/oiled#"
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
        xmlns:xsd="http://www.w3.org/2000/10/XMLSchema#">
        <daml:Ontology rdf:about="">
          <dc:title>"An Ontology"</dc:title> ....  </daml:Ontology>
        <daml:Class rdf:about="file:/C:/HUGO.daml#HUGO">
          <rdfs:label>HUGO</rdfs:label> ...
          <oiled:creationDate> <![CDATA[2003-06-25T07:45:22Z]]>...     </oiled:creationDate>...
        </daml:Class>
        <daml:DatatypeProperty rdf:about="file:/C:/HUGO.daml#OMIM_ID">
          <rdfs:label>OMIM_ID</rdfs:label> ..... </daml:DatatypeProperty>
          <rdf:Description rdf:about="file:/C:/HUGO.daml#HUGO_1">  ....
            <rdf:type>
              <daml:Class rdf:about="file:/C:/HUGO.daml#HUGO" />
            </rdf:type>
            <ns0:HGNC_ID>  <xsd:string xsd:value="247" />  </ns0:HGNC_ID>
            <ns0:GeneName>
              <xsd:string xsd:value="albinism-deafness syndrome" />    </ns0:GeneName>
            <ns0:GDB_ID>  <xsd:string xsd:value="118977" />  </ns0:GDB_ID>
            <ns0:OMIM_ID>  <xsd:string xsd:value="300700" />  </ns0:OMIM_ID>
          </rdf:Description>
        ....
```

Fig. 7. DAML+OIL of Hugo, GDB, OMIM relation

PubMed provides the abstract of relevant literature and also supports it in a XML form with a limitation of document number below to 10,000 items. In order to check the relevancy of each result group and each document, abstracts of each group are extracted into a XML-based input file and estimate the $L(t)$.

5 Empirical Evaluation

We empirically evaluated the system with a simple query and a query with clinical query option in PubMed. Ontology part is implemented with DAML+OIL language and OilEd editor is used partly. The abstracts, author, title, journal, and date information of document are extracted into XML-based input file in order to evaluate the relevancy of documents to a query given. In addition to the Semantic Web languages, the rest of implementation is done in Java.

First of all, we examine both free-text search and MeSH terms for subject heading search separately with a simple query. If a user makes a query with 'deafness' disease name, then PubMed automatically puts the simple query with MeSH terms and a text word such as in **(("hearing loss"[MeSH Terms] OR "deafness"[MeSH Terms]) OR deafness[Text Word])**. It was pointed out by earlier study that free-text searches have a lower sensitivity than subject heading searches, but that the specificity of free-text searching is better than its sensitivity. The study [11] shows that the use of MeSH to

improve sensitivity and specificity relies heavily on high quality consistent indexing. The result of the study showed that the free-text search had a higher both sensitivity and specificity rates than the subject heading search.

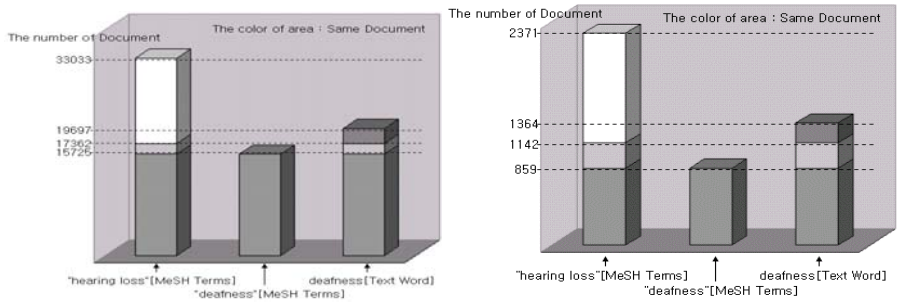


Fig. 8. Subject Heading Search [MeSH] vs. Free-text Search (a) simple query, (b) simple query with clinical query

Our result in Fig. 8 supports the Harrison's study in that a free-text search with 'deafness' word only retrieves 15725 documents out of 33033 documents retrieved by both 'hearing loss[MeSH] term and PubMed query of (("hearing loss"[MeSH Terms] OR "deafness"[MeSH Terms]) OR deafness[Text Word]). The MeSH term search which is a subject heading search picks up some irrelevant articles and many articles on evidence-based health care which are otherwise elusive.

Table 1. The results of different simple query

Query	# of documents (N)	relevancy of each group of documents
Disease (=deafness)	33033*	None
Disease + Genes (OMIM)	771	0.0904
Disease + Genes (Ensembl)	346	0.1281
Disease + Genes (Hugo+GDB+OMIM)	76	0.1216
Disease + Genes (Hugo+GDB+OMIM+LocusLink)	81	0.3558

Table 1 shows the number of documents retrieved by each query and the relevancy of each group of documents retrieved. Here each query means a query submitted to PubMed after pursuing a search strategy of an agent has taken starting from a disease name 'deafness' from a user. The relevancy of each group to a query is evaluated based on overall frequencies of the keyword. A query with a 'deafness' disease name has 33033 documents as the search result and the number of abstracts that we can extract through PubMed is limited to 10,000[4]. Therefore the first query couldn't be evaluated for relevancy checking. Documents retrieved using OMIM references only were 771 and 346 with Ensembl, 76 with Hugo-GDB-OMIM, and 81 with Hugo-GDB-OMIM-locusLink respectively. The relevancy of returned documents was high when Ensembl was utilized.

[4] Message returned from PubMed: can't received file. You tried to save 33033 items. The maximum number of items that can be saved is 10000. Refine your search or select specific items.

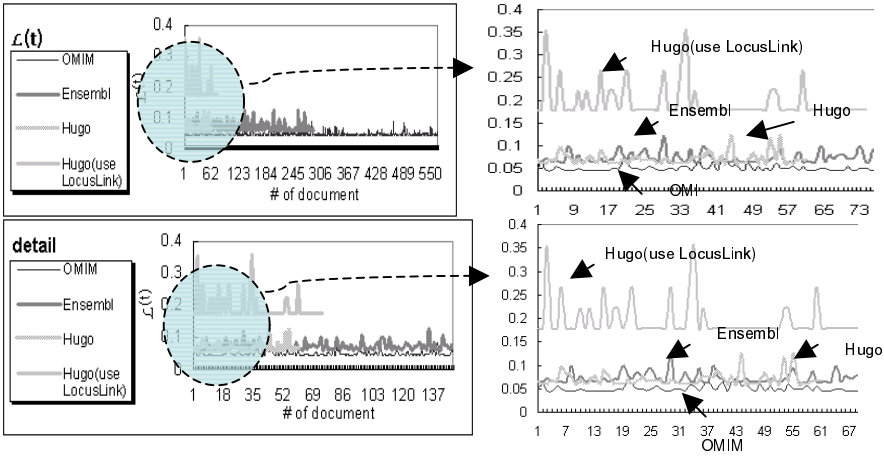


Fig. 9. The result of relevancy of each document in each group

The relevancy of each document retrieved in each group is evaluated with *L(t)* in Section 4. Each search strategy taken by individual information agent is denoted as OMIM, Ensembl, Hugo, or Hugo(locusLink) in Fig. 9. Here Hugo denotes Hugo-GDB-OMIM and Hugo(LocusLink) denotes Hugo-GDB-OMIM-LocusLink. As seen in Fig. 10, the result for relevancy checking of each document indicates that using Ensembl outperformed and using Hugo-GDB-OMIM followed closely while using OMIM only had the lowest performance among the three. Using PubMed only couldn't be evaluated due to too many documents retrieved.

We experimented the same simple queries with clinical query provided by PubMed interface: therapy, diagnosis, etiology, and prognosis. The quality of the results returned is also examined with emphasis: sensitivity and specificity. The query with etiology clinical query showed the highest sensitivity among other clinical query: therapy, diagnosis, and prognosis and second highest specificity. The query with diagnosis had the highest specificity but its sensitivity couldn't be evaluated due to the excess of limiting number of documents to be extracted with 12946 documents. We further examined the performance of search strategies mentioned for each query with different clinical query option and emphasis option. In terms of optimal balance of sensitivity and specificity, a query with etiology clinical query had the best result and we used the result to evaluate search strategies over other results as a representative here. As in Fig. 10, other search strategies in general outperformed to when only PubMed is used. The performance was Ensembl, OMIM and PubMed in order. With clinical query, Hugo-GDB-OMIM returned nothing most of eight queries. When Hugo-GDB-OMIM-LocusLink retrieved documents such as in therapy, diagnosis, and etiology, it showed the highest relevancy.

The search results were extracted into XML-based data and evaluated for relevancy with raw data form MEDLINE instead of using a gold standard database that is often used in biomedical study to test the revised strategy in PubMed format. The attempt to weave through different search engines for finding correlated information was tried and achieved better performance than using a PubMed interface only for document retrieval. Nevertheless, we've learned that the results of searches

can be dependent on a number of different variables. Expertise of searcher, consistency of indexer, and the nature of the subject of the information query all influence on the results. The study on systematic analysis of indexing schemes through out search engines and ontologies needs to be done further.

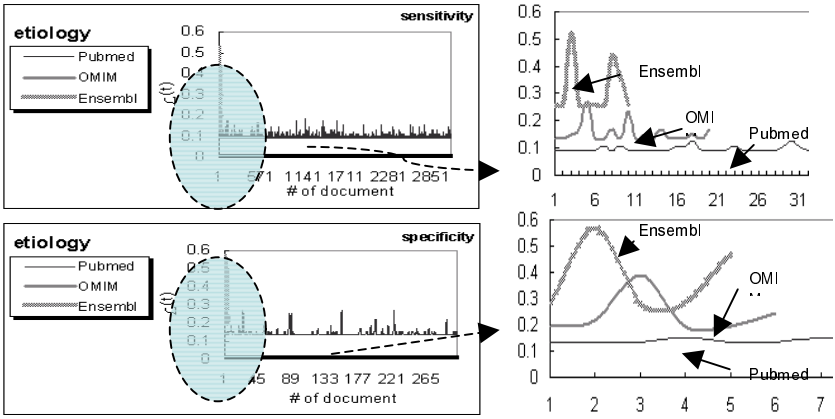


Fig. 10. The result of a query with clinical query of etiology

6 Related Work

[12] MELISA (Medical Literature Search Agent) is another representative ontology-based information retrieval system. Our work is closely similar to the work in sharing the goal to achieve and using ontology. MELISA demonstrates how ontologies can be very useful in enhancing Web searches. It is based on subject heading search using MeSH terms, while our work is more based on free-text search employing distributed agents for using correlations among biomedical databases available on-line. [13] is another information retrieval system using both free text and semantically enriched markup as our system is. Its inference engine is more general than ours in that DAMLJessKB is built on Jess which uses the Rete algorithm, while ours has our own inference engine to process rules.

7 Future Work

To provide an ultimate quality and performance with unlocking large amounts of (un)structured data, the factors of performance and human approach besides specificity and sensitivity, (that is, not only documents but also relevant experts and organizations), should be shown as a result of a search. Therefore the use of specific author name or organization name will be examined further to improve both the sensitivity and specificity. More functions are to be developed to weigh terms in a query depending on what a user puts and how close a term derived from an ontology

to a query given. Methods of coordinating and combining results returned by individual information agents are under development.

Other more advanced search engines are based on self-learning principles. But how well these systems learn and how they learn correctly is important so therefore, we are examining self-learning algorithm such as Bayesian network for agent ontology. Further we will attempt to include user profiles in order to help users by providing correlated information through user's individual interests and/or genetic inheritances.

Acknowledgement. This work was supported by grant No.(R05-2002-000-01351-0) from Korea Science & Engineering Foundation.

References

[1] J. Sowa, Knowledge Representation: Logical, Philosophical, and Computational Foundations. New York:PWS Publishing Co., 1999
[2] M. Uschold and R. Jasper, A Framework for Understanding and Classifying Ontology Applications, In Proceedings of the IJCAI-99 Workshop on Ontologies and Problem-Solving Methods.
[3] P. Karp, An Ontology for biological function based on molecular interactions, Bioinformatics 16(3):269–285, 2000
[4] Semantic Web in W3C http://www.w3c.org/2001/sw
[5] D. Hunt and K. McKibbon, Locating and appraising systematic reviews., Ann Intern Med. 1997;126:532–8
[6] J. Glanville and C. Lefebvre, Identifying systematic reviews: key resources ACP J Club. 2000; 132:A11–2
[7] L. Bero and A. Jadad, How consumers and policymakers can use systematic reviews for decision making, Ann Intern Med. 1997; 127:37–42
[8] A. Eberhart, An Agent Infrastructure based on Semantic Web Standards, submitted to Elsevier Science, May 2002
[9] H. Boley, S. Tabet, G. Wagner, Design rationale of RuleML: A markup language for Semantic Web rules, in Semantic Web Working Symposium, 2001.
[10] TRIPLE Semantic Web, http://triple.semanticweb.org/, March, 2002
[11] J. Harrison, Designing a search strategy to identify and retrieve articles on evidence-based health care using MEDLINE, Health Library Review, 14(1):33–42, Mar. 1997.
[12] J. M. Abasolo and M. Gomez, MELISA. An ontology-based agent for information retrieval in medicine, Semantic Web Proceedings in 2002
[13] U. Shah, T. Finin, A. Joshi, R. S. Cost, and J. Mayfield, Information Retrieval on the Semantic Web, submitted to 2nd Conf. Semantic Web, 2003

Agent-Based Intelligent Clinical Information System for Persistent Lifelong Electronic Medical Record

Il Kon Kim[1] and Ji Hyun Yun[2]

[1] Il Kon Kim, Computer Science, Kyungpook National University,
1370 Sankyuk-dong Buk-ku, Daegu, Korea
ikkim@knu.ac.kr
http://giant.knu.ac.kr
[2] Ji Hyun Yun, Computer Science, Kyungpook National University,
1370 Sankyuk-dong Buk-ku, Daegu, Korea
yjh9626@cs.knu.ac.kr

Abstract. Hospital systems are heterogeneous and the clinical information scattered. The development of a clinical information system to integrate the information among hospitals and make legacy systems cooperate is thus difficult. For efficient clinical information management and system integration, clinical data, patient data privacy and integration of records need to be well organized and conform to relevant standards. In this paper we develop an agent-based intelligent clinical information system for persistent lifelong electronic medical record. Functional entities are divided according to tasks and implemented as collaborating agents. These agents reside on a multiagent platform which provides communication and invocation of execution functionality. We adopt HL7 standards for both clinical data and web services, in order to provide a user-environment via the internet.

1 Introduction

Many healthcare institutions are seeking to develop integrated clinical workstations - single-entry points into a medical world in which computational tools assist not only with clinical matters (reporting results of tests, allowing direct entry of orders by clinicians, facilitating access to transcribed reports, and in some cases supporting telemedicine applications or decision-supporting functions) but also with administrative and financial topics (tracking of patients within the hospital, managing materials and inventory, supporting personnel functions, managing the payroll), research (analyzing the outcomes of associated trials, and implementing various treatment protocols), scholarly information (accessing digital libraries, supporting bibliographic search, and providing access to drug information databases), and even office automation (providing access to spreadsheets). The major issues with medical records are their electronic format, accessibility, sharability, confidentiality, security, acceptability to clinicians and patients, and integration with other types of nonpatient-specific information.

Despite the fact that traditional paper-based medical records are EMRs (electronic medical records), the development of effective EMRs has been hampered by (1) the

J. Lee and M. Barley (Eds.): PRIMA 2003, LNAI 2891, pp. 194–204, 2003.

need for standards in the area of clinical data, (2) concerns regarding data privacy, confidentiality, and security, (3) challenges of data entry by physicians, and (4) difficulties associated with the integration of record systems with other information resources in the healthcare setting.

In this paper we propose an agent-based intelligent information system for effective and interoperable clinical support of HL7 interface engine to the goal of lifetime persistent electronic medical record. We adopt the HL7 (Health Level 7) standard to create a health care messaging standard and show the design and feasibility of the resulting system architecture. In order to integrate into a heterogeneous hospital system, we develop a variety of functions such as data exchange, data management, and data integration that support clinical patient care as agents based on a multiagent platform. For evaluation of health care services via the Internet we use a web service technique based on .NET platform.

2 Sharing Medical Information

Health-care records are widely distributed and have strong local autonomy [1]. Frequently, patient information is not available when and where it is needed. This lack results in multiple requests to patients for personal data, unnecessary duplication of tests and other investigations, and ultimately delays in the patients receiving appropriate care [2]. Synapses [3] and SynEx [4] developed as part of two European Commission-funded projects which share electronic health-care records through the internet and the web. Synapses concentrated on the specification of a Federated Healthcare Record (FHCR) server, which provides integrated access to a record's distributed components. The Synapses server provides the functionality of a federated database system [5]. Individual patient data could be distributed across numerous hospital and community health sites, which might necessitate forwarding a query to a server located elsewhere. A Synapses server at an external site would then act as a federating device, hiding the details of its local feeder systems from the remote.

The SynEx project concerned integrating a number of components – the Synapses record server being just one – to form an information system from which client applications could access a wide range of data in support of the health-care business. In SynEx, the record server was further developed as a component within an open, distributed-computing environment in an enterprise-wide manner and using XML technologies, with support for Web-based clients.

On the other hand, the recognition of the need for interconnection led to the development and enforcement of data-interchange standards, which include a standard medical vocabulary, standards for hospital information systems, standards for the computer-based patient record, and standards for data interchange. As a result, an integrated hospital information system was developed by interconnecting function-specific systems. The name HL7 reflects the seventh (application) level of the OSI reference model [6]. The primary goal of HL7 is to provide a standard for the exchange of data among hospital-computer applications.

The HL7 standard is message based and uses an event trigger model that causes the sending system to transmit a specified message to the receiving unit with a subsequent response by the receiving unit. Messages are defined for various trigger events. Version 1.0 was published in September 1987, and on February 8 1996, it was

approved by ANSI as the first health care data-interchange standard. The HL7 organization is also developing standards for the representation of clinical documents (such as discharge summaries and progress notes). These document standards make up the HL7 Clinical Document Architecture (CDA). The HL7 CDA Framework became an ANSI-approved HL7 standard in November 2000. The CDA is a document markup standard that specifies the structure and semantics of clinical documents. A CDA document is a defined and complete information object that can include text, image, sounds and other multimedia content and gateway for Electronic Medical Record to support sharing and interoperability of Electronic Health Record.

For consistency, we adopt HL7 standards in the present project. Designing the system from the small and light software program to the large system architecture, we chose to use the OOP concept. Object oriented programming techniques have lots of advantages, especially the possibility of developing systems composed of individual working objects. An even better approach is to adopt the agent concept. An agent is a computational entity such as a software program or a robot that can be viewed as perceiving and acting upon its environment, and that is autonomous in that its behavior, at least partially, depends on its own experience [7].

Many other different modules are made with autonomous agents, with each agent composing the system and working together dynamically - this is referred to as "Interacting". A multiagent system primarily focuses on coordination as a form of interaction, which is particularly important with respect to goal attainment and task completion. The purpose of coordination is to achieve or avoid states that are considered as desirable (or undesirable) by one or more agents. To coordinate their goals and tasks, agents have to explicitly take dependencies among their activities into consideration [8]. Cooperating agents try to accomplish as a team what individuals cannot, and so fail or succeed together.

To build our system in which agents interact appropriately, the following issues need to be addressed.

- How to enable agents to communicate. What communication languages and protocols to use.
- How to enable agents to represent and reason about the actions, plans, and knowledge of other agents in order to appropriately interact with them.
- How to enable agents to find out whether they have achieved progress in their coordination efforts, and how to enable them to improve the state of their coordination and to act coherently.
- How to enable agents to form and solve organizational structures.
- How to wrap proprietary component systems via agents
- How to formally describe multiagent systems and the interactions among agents.
- How to integrate distributed clinical document repositories like a intelligent health information system and get its performance of very large XML distributed databases.

A multiagent system can be effectively used to construct a system for supporting clinical information. For agents to communicate and interact in a multiagent system, HL7 standard can be used. With these two of mutual benefit we concentrate on develop the system efficiently, so our research can be very remarkable in the point of view of the most recent study.

3 Agent-Based Intelligent Clinical Information System

Firstly let's imagine the Fig. 1 scenario. The patients can access their previous clinical record on the internet, if a clinical data repository exists.

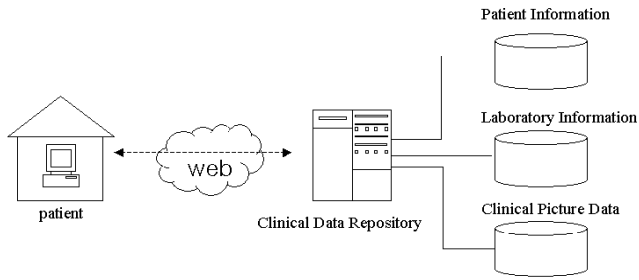


Fig. 1. The first scenario. Patients can access their clinical history with a computer in home or office. The clinical data repository processes the request to make individual queries into each specific system that store the patient data. The patient information system, laboratory information system, and clinical picture data system process these requests and pass the responses back to it. The clinical repository integrates the responses and returns the results to the patient

The clinical data repository integrates individual patient records and comprises several component systems, which are the patient information system, the laboratory information system and the clinical information system. The clinical document architecture generates XML based clinical documents and these clinical documents added HL messaging interface protocol are stored into clinical data repository as document centric data of native XML databases.

An electronic healthcare record is a structured multimedia collection of health-care data about an individual patient. Extracts of individual patient records, obtained from a system and brought together in a structured format, can be shown to the patient. Figure 2 shows the second scenario.

In hospitals and other distributed health care environments, myriad data needs to be collected by multiple healthcare professionals who work in a variety of settings; each patient receives care from a host of providers-nurses, physicians, technicians, pharmacists, and so on. Communication among the members of the team is essential for effective healthcare delivery. Communication among the different hospitals is often also required. Data must be available to decision-makers when and where they are needed.

Furthermore, in many institutions, clinical and financial activities are supported by separate organizational units. Hospital administrators must integrate clinical and financial information to analyze costs and to evaluate the efficiency of healthcare delivery. Similarly, clinicians may need to review data collected at other healthcare institutions, or they may wish to consult online databases of biomedical information. Local-area networks that permit the sharing of information among independent computers and wide-area networks that permit the exchange of information among geographically distributed sites provide good communication infrastructures. The actual integration of information requires additional software, adherence to standards, and operational staff to keep it all working as technology and systems evolve.

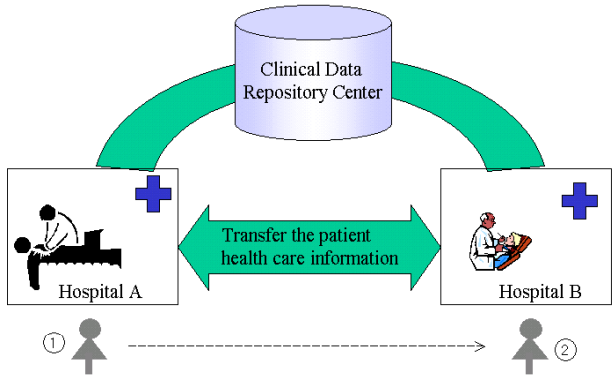


Fig. 2. The second scenario. A patient from hospital A transfers to hospital B. In hospital B, the doctor may need the past clinical record for diagnosis. In this case, if the patient record in hospital A has been stored in a clinical data repository, hospital B requests a transfer of the patient's clinical information and receives it. Alternatively, hospital B directly requests hospital A to send the clinical information

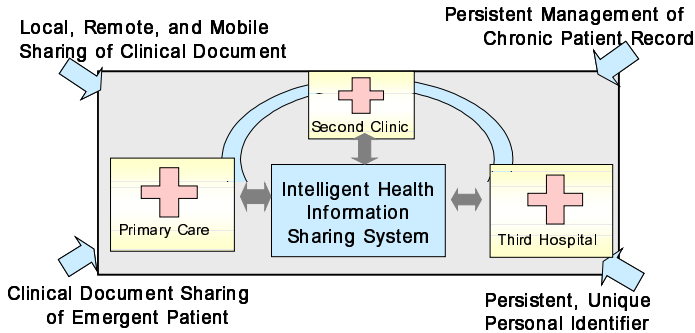


Fig. 3. Intelligent Health Information Sharing System

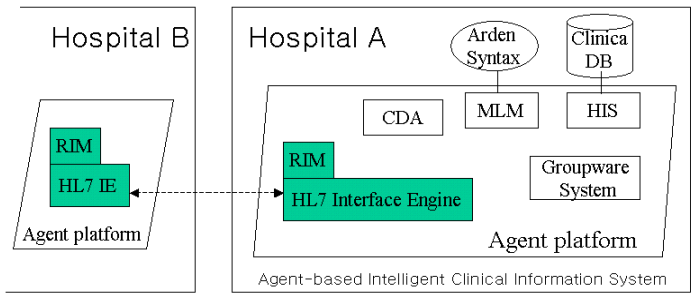


Fig. 4. Inter-hospital communication through the HL7 interface engine RIM

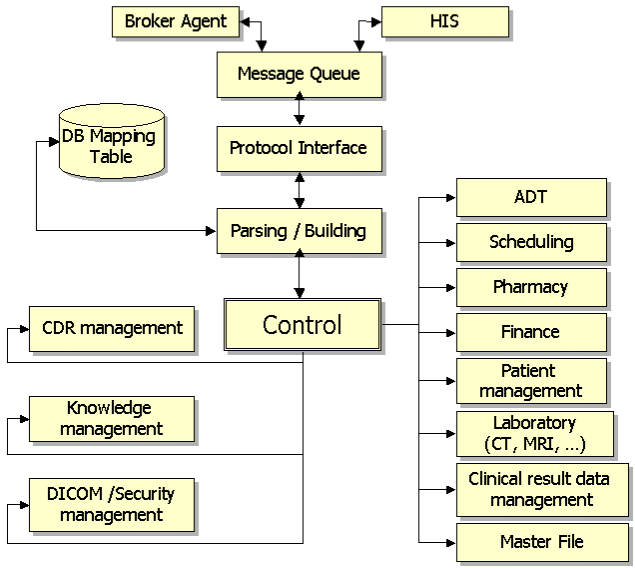


Fig. 5. HL7 interface Engine

Figure 3 shows an intelligent Health Information Sharing System which has 4 objectives, Local, Remote, and Mobile Sharing of Clinical, Persistent Management of Chronic Patient, Clinical Document Sharing of Emergent Patient, Persistent, Unique, Personal Identifier management.

Figure 4 shows inter-hospital communication using the HL7 interface engine. The HL7 interface engine basically supports the data integration in infra environment, and is sharing the HL7 RIM (Reference Information Model).

The Agent-based Intelligent Clinical Information System is composed of several entities as shown in Fig.4. These are the HL7 interface engine and the CDA (Clinical Data Architecture), MLM (Medical Logic Module), HIS (Hospital Information System), CCOW (Clinical Context Object Workgroup) and Groupware system. The HL7 interface engine permits the message parsing and construction, and also sends control management message to the system.

Figure 5 shows the HL7 interface engine internal entities. The important specific functions are as follows.

- Message queue
  - input the message from the Broker Agent, HIS
  - send the message into the Protocol Interface
- Message mapping in referencing the DB Mapping Table
  - generate the message by fitting the message into the HL7 message standard form with composing fields
  - manage and store the mapping table information separately
- Message Parsing
  - analyze the message XML tag by referencing the message mapping table
  - make action message work according to the message type
  - store the message in the Database to record history

- Message Building
  - build the HL7 message by referencing the message the mapping table
  - use the autonomous building function without the user directed controlling
- Message Control
  - execute the action which the message actually wants, according to the message type, ADT (Admission-discharge-transfer), scheduling, pharmacy, laboratory information management etc.

The CDA is part of the HL7 version 3 family of standards. This family, which includes both CDA and the evolving version 3 message standards, all derive their semantic content from the shared HL7 Reference Information Model (RIM) and are implemented in eXtensible Markup Language (XML). The process for generating an XML-based implementation from the RIM is part of the version 3 development methodology [9]. The exact style of HL7's XML representation was a carefully considered balance of technical, practical, and functional considerations.

A CDA document is a defined and complete information object that can include text, images, sounds, and other multimedia content. The document can be sent inside an HL7 message and can exist independently outside a transferring message [10]. There is a critical interdependence between clinical documents and document management systems. If CDA documents are viewed outside the context of a document management system, it cannot be known with certainty whether or not the viewed document is current. Clinical documents can be revised, and they can be appended to existing documents. In practice, it is impossible to guarantee an explicit forward pointer from an outdated version to the newer version. Document management systems and HL7 messages that carry CDA documents convey critical contextual information that minimizes the risk of viewing superseded information.

The Medical Logic Module (MLM) contains sufficient knowledge to make a single decision. It is an independent unit in a healthcare knowledge base. Each MLM contains maintenance information, links to other sources of knowledge, and enough logic to make a single decision. The health knowledge contains contraindication alerts, management suggestions, data interpretations, treatment protocols, diagnosis scores etc. For example, when a doctor prescribes the patient penicillin, the MLM checks for contraindications. If the patient has a recorded allergy to penicillin, an alert is generated. This decision support system in clinical institutions has been used successfully for many years, and is useful in preventive medicine. The syntax of each knowledge base is different and cannot be shared. Accordingly, Arden syntax attempts to define a complete health knowledge base for clinical decision support [11]. The HL7 standard interprets it as knowledge of the decision support. An MLM with Arden syntax can be used to share knowledge in a heterogeneous system, enabling common clinical treatment can be made.

With an emphasis on the point-of-use of applications, CCOW is to define standards that enable the visual integration of healthcare applications. Applications are visually integrated when they work together in ways that the user can see in order to enhance the user's ability to incorporate information technology as part of the care delivery process. CCOW has focused on specifying the Context Management Standard. Context management entails the coordination and synchronization of applications so that they are mutually aware of the set of common things - known as the context - that

frame and constrain the user's interactions with applications. The context is primarily comprised of the identity of real-world things, such as patients, and real-world concepts, such as encounters, that establish the common basis for a consistent set of user interactions with a set of healthcare applications. The elements of the context may be provided directly by users as they interact with applications, or may be provided automatically by underlying programmatic sources. Specifically, the Context Management Standard defines a protocol for securely "linking" applications so that they "tune" to the same context. The context is represented as a set of subjects, each of which generally identifies a real-world entity such as a patient or a real-world concept such as an encounter. Linked applications remain automatically synchronized even when a context subject changes, for example, due to the user's inputs (e.g., the user selects a different patient). The user inputs of particular interest are those that are used to identify something, such as medical record number for patients, or an account number for a clinical encounter. CCOW specifies technology-neutral architectures, component interfaces, and data definitions as well as an array of interoperable technology-specific mappings of these architectures, interfaces, and definitions. It is the intent of CCOW that its standards may be implemented using broadly accepted and/or industry standard computing technologies, with emphasis on those that are most relevant to the HL7 membership. It is also the intent of CCOW that its specifications provide all of the details needed to ensure robust and consistent implementations of compliant applications and system services. However, CCOW does not intend for its specifications to serve as an implementation or the design of an implementation[6].

The hospital information system supports a variety of functions, including the delivery and management of patient care and the administration of the health organization. From a functional perspective, the HIS consists of components that support the following distinct purposes: (1) patient management, (2) departmental management, (3) care delivery and clinical documentation, (4) financial and resource management and (5) managed-care support [12]. For this legacy system to be used in a web environment, we add a groupware system. This system enables physicians to manage each component effectively and make it possible to use it on Internet. Usually a groupware system is user-help software, and HIS has some those functions too, without regard to the environment of using Internet the groupware system is added.

In our system the various described entities are actually delegate agents, and they are on a multiagent platform. Figure 6 shows the detailed structure.

Each entity is implemented as a delegate agent, which has a specific function. Actually each agent resides on a multiagent platform. Among heterogeneous systems, it can be possible to communicate and share the clinical information [13–15].

Firstly, an agent registers in the AMS (Agent management system). The AMS is in charge of the management of agent states. When an agent is in the platform and it registers with the AMS, it is serviced in the platform. In other words, agents communicate with the other and use the resources offered by them. Similarly when it is out of the platform and deregisters the AMS, it is not an agent in the platform. Other agents cannot work together with it and request help.

As the name MTS (Message Transport System) implies, the MTS provides the way that the agent sends and receives the message. Agents are either in the same platform or not, if agents are registered in the platform, they can communicate through MTS, because MTS guarantees platform transparency.

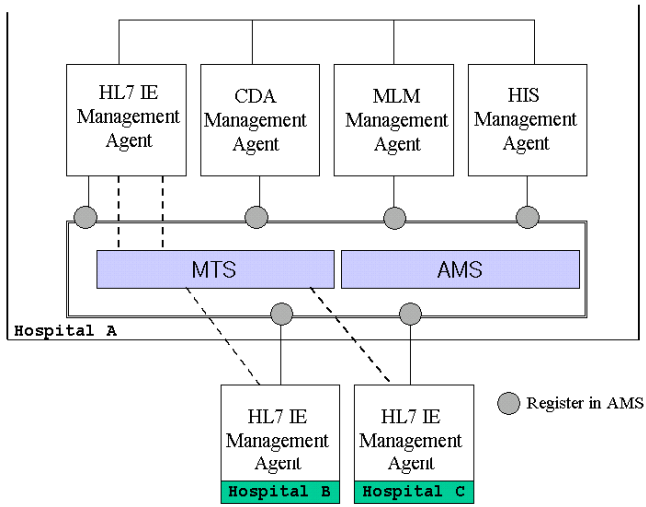


Fig. 6. The internal structure of the agent-based intelligent clinical information system. Each entity described in fig.5 is implemented as an agent, which is delegated a specific function and each agent is on a multiagent platform. In this platform, individual and heterogeneous systems can communicate and share the clinical information.

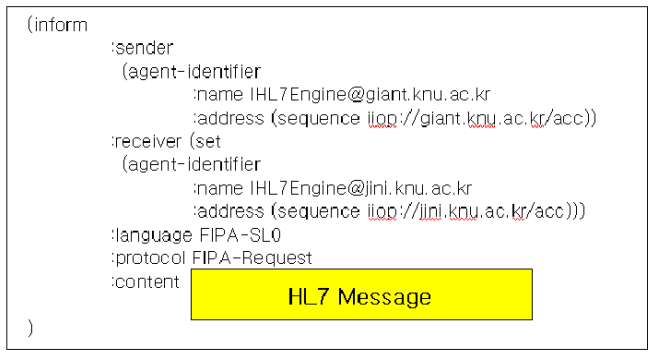


Fig. 7. An ACL message containing a HL7

Fortunately ACL (Agent Communication Language) can be encoded with XML like the HL7 3.0 message format. One agent can send other ACL messages that contain the HL7 message in the content tag. A typical example is shown in Fig. 7.

Each message token just like “inform” and “sender” is encoded in the XML tag where it is possible to parse using the XML parser. The ACL message containing the HL7 message is sent to other agents in its platform or another platform, an agent who receives the message can then execute the specified actions.

The agents registered in the multiagent platform are co-workers. This framework is well fitted in the clinical system because the system wants to send and receive the clinical data, show the records to the users, update the in the database dynamically and use the records for decision support, and so on. Usually agents in this system

collaborate but they compete with each other when using record in the same database. The well-composed multiagent platform controls situations and manages them efficiently.

In this paper we use a "multiagent platform" for the development of "clinical information system". In this system the actual function is implemented as agents, and the physical environment is agents' residing in a multiagent platform. As described above, the multiagent platform is well suited to a clinical information system because it supports a dynamic physical environment. When a hospital clinical information system needs patient records from another system, the HL7 interface engine management agent becomes the messenger. In other words, the HL7 interface engine management agent registers the other system by sending an ACL registration message, and through the MTS it can actually communicate with target agents in the other clinical information system.

4 Implementation Focus

Web services of Microsoft are the latest thing in application development and have attracted the interest of many developers. The fundamental concept is simple; web services allow us to make Remote Procedure Calls (RPCs) against an object in the Internet or a network. Web services differ from previous technologies in that their use of platform-neutral standards such as HTTP and XML allows us to hide the implementation details entirely from the client. The client needs to know the URL of the service, and the data type used for the method calls, but doesn't need to know whether the service was built in Java and is running on Linux, or is an ASP.NET web service running on Windows [16].

In this environment we regard clients as either HL7 interface engine management agent, the patient users or other clinical institutions. Clients could discover a web service and access it when the service function has been built in to the system. In the inner system of clients, the methods create a proxy class for accessing a web service via HTTP-GET or HTTP-POST, or using the SOAP method. This makes accessing a remote service as simple as using a function in a local-client DLL file.

As web services work based on XML and HTTP or SOAP, it is appropriate that our system have the advantage of being accessed and developed simply. To locate web service providers, we use UDDI (Universal Description Discovery and Integration) specifications, which define XML Schemas that provide information about web services that allow clients to consume web services.

We suggest implementing agents in the .NET framework of Microsoft. This provides a useful and easy user-interface development environment, and contains a variety of utilities for developing web services.

5 Discussion and Conclusion

In this paper we have shown the architecture, entities' function, development and implementation of an agent-based intelligent clinical information system.

For effective communication among the hospital or clinical institutions, sharing of clinical data records for decision support and providing the records to the patients, cooperative attitudes of common participants are needed. Surely there exist potential difficulties because many legacy systems are heterogeneous and the integration of these systems would not be easy. To solve this problem we propose the agent-based intelligent clinical information system using the HL7 standards, intelligent agent concepts based on multiagent platform, and a web service development technique provided by Microsoft by .NET platform.

Focusing on the expansion and reusability of the system, we abide by the national-standards of each study area. The clinical information system needs to be supported by the dynamic environment management, which the multiagent system can support. Furthermore, for patients, clinicians and physicians to use this system easily and effectively via the internet, a web-based system must be established. The latest development technology in this regard is a web service based on the .NET framework.

Acknowledgement. This study was supported by a grant of the Korea Health 21 R&D Project, Ministry of Health & Welfare, Republic of Korea (02-PJ1-PG6-HI03-0004).

References

1. Networking Health; Prescriptions for the Internet. Tech report TCD-CS-2001-01, Dept. of Computer Science, Trinity College Dublin; http://www.cs.tcd.ie/publications/tech-reports/.
2. R.S. Dick et al, The Computer Based Patient Record: An Essential Technology for Healthcare, National Academy Press, Washington, D.C., 1997
3. Synapses; http://www.cs.tcd.ie/synapses/public
4. SynEx; http://www.gesi.it/synex
5. W. Grimson et al., "Federated Healthcare Record Server – The Synapses Paradigm", International Journal of Medical Informatics, vol.52, 1998, pp. 3–27
6. HL7; http://www.hl7.org/
7. G. Weiss, Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence, MIT press, Cambridge, Massachusetts, London, England, 2000
8. M.N. Huhns et al, Reading in Agents, Morgan Kaufmann Publishers, Inc., San Francisco, California, 1998
9. G.W. Beeler. "Taking HL7 version 3: an object-oriented methodology for collaborative standards development", International Journal of Medical Informatics, vol.48 (1-3), 1998, pp. 151–61
10. R. H. Dolin et al, "The Practice of informatics - The HL7 Clinical Document Architecture", International Journal of Medical Informatics, vol. 8, 2001, pp. 552–561
11. http://www.cpmc.columbia.edu/arden/
12. E.H. Shortliffe et al, Medical Informatics: Computer Applications in Health Care and Biomedicine, Springer-Verlag, New York, 2001
13. M. d'Inverno et al, Understanding Agent Systems, Springer-Verlag, Berlin Heidelberg, 2001
14. http://www.fipa.org
15. N.R. Jennings et al, Agent Technology, Springer-Verlag, Berlin Heidelberg, 1998
16. A. Banerjee et al, C# Web Services, Wrox Press, Birmingham, UK, 2001

Extended Hierarchical Task Network Planning for Interactive Comedy

Ruck Thawonmas, Keisuke Tanaka, and Hiroki Hassaku

Intelligent Computer Entertainment Laboratory
Department of Computer Science, Ritsumeikan University
Kusatsu, Shiga 525-8577, Japan
ruck@cs.ritsumei.ac.jp

Abstract. We focus on interactive comedy, a relatively new genre in interactive drama, in this study. To be able to maintain the story, we adopt hierarchical task network planning (HTN) for planning of a character agent. We then extend HTN such that the extended HTN allows for the direct use of character failures, higher interactions with the viewer, and more story variations. Our extensions lead to effective extraction of laughter from the viewer. We test our system with a short visual comedy similar to the popular Mr. Bean^{TM} series.

1 Introduction

Interactive dramas are a new type of next generation digital contents in which a viewer can interactively participate in the story. Research on interactive drama has been recently very active [1-4,7-9]. Here we divide interactive drama systems into two categories according to the perspective of the viewer, namely, the first-person perspective and the third-person perspective. An example of the former system is *Facade* [1] where the viewer participates in the story as one of the character agents and his or her interactions with the other agents can change the ending of the story. An example of the third-person-perspective systems is a system developed at University of Teesside [2] where the viewer participates in the story from the third-person perspective, but can interact with the system through manipulating items, such as hiding a diary, and giving voice-based advice to a character agent of interest.

The interactive drama system that we develop is a third-person-perspective comedy system. In our system, a character agent unfolds the story while allowing interactions with the viewer. Such interactions lead to humorous behaviors of the character agent. Existing interactive comedy systems include a system developed at ATR [3] that focuses on humorous conversation or dialogue between the viewer and the agent. However, there is no concept of story development in that system. Another system developed at University of Teesside [4] considers both humor and story. However, in that system, humorous situations are caused by an uncontrolled failure to perform a task or sub-task of a character agent, and thus are limited. As shall be seen in Section 3, our system can generate more

J. Lee and M. Barley (Eds.): PRIMA 2003, LNAI 2891, pp. 205–213, 2003.

humorous situations by extracting laughter from the viewer with controlled failures.

In this paper, we propose the method for planning of a character agent in our interactive comedy system. Planning is an important research issue and relates directly to story development in interactive drama. We compare hierarchical task network planning (HTN) [2, 5] and heuristic search planning (HSP) [4, 6]. Both of them have been applied to existing interactive drama systems. We eventually adopt HTN because it allows us to maintain the story more readily than HSP. However, HTN is known to provide limited variations in the story. With the aim to effectively extract laughter from the viewer, the proposed method extends HTN to allow for the direct use of character failures, higher user interactions, and more story variations.

The organization of the rest of the paper is as follows: Section 2 gives an introduction of HTN and HSP. Section 3 discusses our extensions to HTN. Section 4 describes our early results. Finally, Section 5 concludes the paper by summarizing the main results and suggesting future work.

2 HTN and HSP

HTN and HSP are planning techniques that have been already applied in interactive drama systems [2, 4]. Their comparisons with respect to interactive drama viewpoint have been discussed recently in [7].

HTN decomposes a given task into a set of smaller sub-tasks, and in the end into sub-tasks called primitive tasks. Task decomposition is represented by a tree constructed prior to planning. The story can thus be maintained by designing the tree accordingly. A plan is generated by performing a search from the top of the tree [5] (cf. Fig. 1). In Fig. 1, the root node of the tree represents the task of interest; the leave nodes are primitive tasks; and the remaining nodes are called methods, each method for a set of related sub-tasks. In HTN, backtracking is performed when execution of a primitive task of interest fails. In general, backtracking is usually seen at low levels of the tree. This causes HTN to provide only limited variations in the story.

HSP, on the contrary, dynamically generates a plan that achieves a given goal. Compared to HTN, HSP thus provides higher story variations. However, it is harder to maintain the story with HSP.

We want to develop a third-person-perspective interactive comedy system where the story is maintainable. Due to this requirement, we adopt HTN. With HTN, interactions with the viewer are done at the primitive-tasks level. We discuss in the next section our extensions to HTN at this level.

3 Our Extensions to HTN

In this section, we propose the following extensions to HTN at the primitive-tasks level.

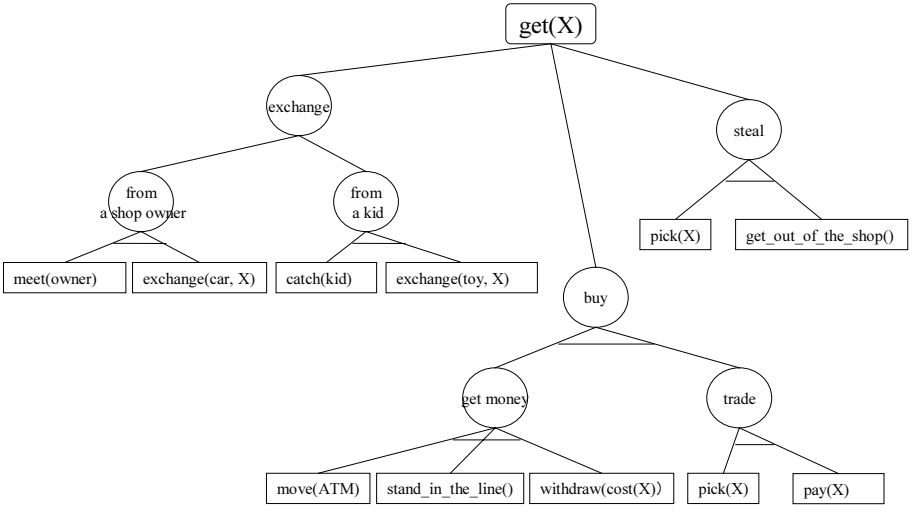


Fig. 1. Example of task decomposition for task **get(X)** in traditional HTN

- Introduction of executability conditions to each operator of a primitive task
- Introduction of multiple candidates in a primitive task, those candidates giving the same effects but in different manners, and a mechanism for selecting them.

3.1 Executability Conditions

In [4], it was discussed that a failure of a drama character to perform a task might lead to a humorous situation, and the concept of executability conditions was proposed and applied to HSP. Adopting a similar idea, we discuss in this section our extension to HTN.

In traditional HTN, a primitive task is described by a STRIPS-style representation as shown in Fig. 2. Under this description, a primitive task is represented by an operator (notated by Operator) consisting of Preconditions, Delete List, and Add List that are the condition(s) that must hold in order to execute the operator, the state(s) to be deleted from the working memory, and the state(s) to be added to the working memory, respectively. Backtracking is performed when the operator of interest has unsatisfied Preconditions. When all conditions given at Preconditions hold, the operator is executed and successful execution is guaranteed, enabling deletion and addition of the states given at Delete List and Add List, respectively.

Our extension here is the introduction of executability conditions for describing condition(s) for success or failure in execution of the operator. Note the difference between **pick()** and **pick_secretly()** or that between **pay()** and **pay_in_a_hurry()** in Fig. 3. Same as in traditional HTN, when all conditions given at Preconditions hold, the operator is executed. However, whether this

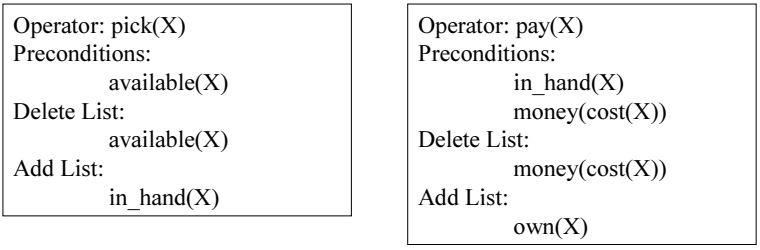


Fig. 2. Example of operators in traditional HTN

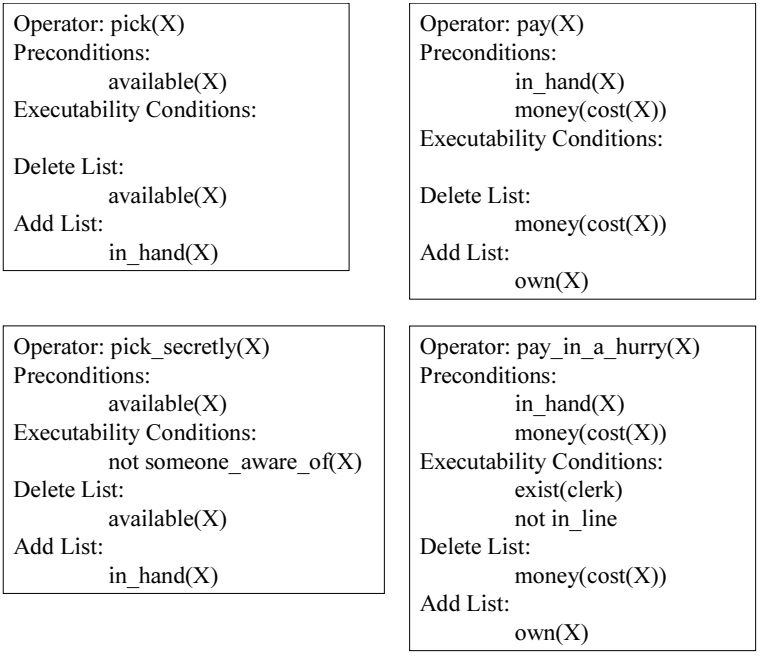


Fig. 3. Example of operators in the extended HTN with the introduction of Executability Conditions

operator execution succeeds or fails depends on the conditions given at Executability Conditions.

In [4], conditions given at Executability Conditions were not used for operator selection. As a result, humorous situations are caused by uncontrolled failures in execution of selected operators, and are hence limited. To have more humorous situations, we argue that those conditions at Executability Conditions should also be used for action selection. In the following section, we discuss our next extension to HTN that copes with this issue.

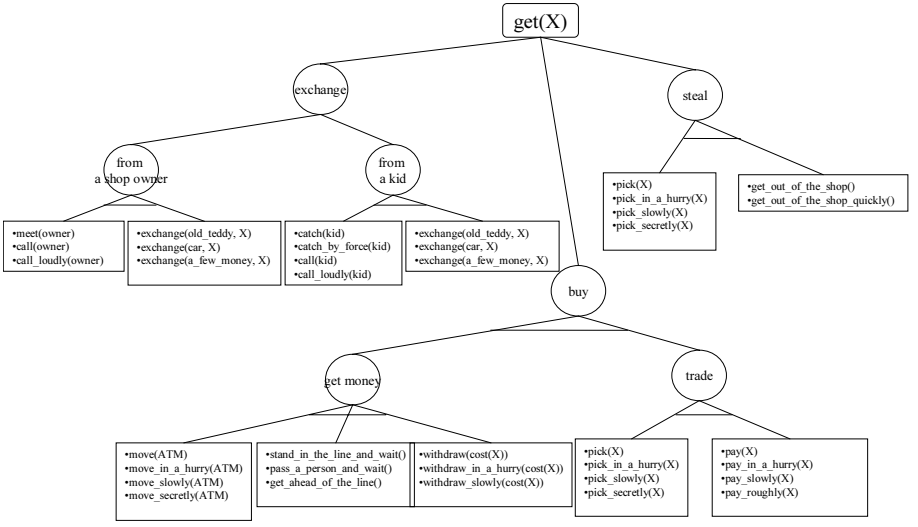


Fig. 4. Example of task decomposition for task **get(X)** in the extended HTN with the introduction of multiple candidates in each primitive task

3.2 Multiple Candidates in a Primitive Task

Our extension to HTN here is that of allowing for multiple candidates or operators in a primitive task, as shown in Fig. 4. All candidates in each primitive task share the same Preconditions, Add List, Delete List, but have different operators and Executability Conditions. Due to sharing the same Preconditions, Add List, and Delete List, selection of the operators, in the same primitive task of interest having satisfied Preconditions, can be arbitrarily done. If one adopts traditional HTN representation and attempts to increase the story variations at one point by adding more primitive tasks whose Preconditions, Add List, Delete List are different, he or she has to examine those primitive tasks one by one whether they have interaction with the primitive tasks in other parts of the tree in order to maintain task decomposability [7]. The extended representation presented in this section therefore makes it easier to increase story variations while requiring no additional efforts to keep task decomposability hold.

In our system, a heuristic function is used to select an operator to be executed from multiple candidates. The heuristic function in use includes components such as mood of the corresponding character agent, satisfaction or unsatisfaction of Executability Conditions of the candidates, and humor-level attached with the failure to execute each candidate. The operator having the minimum heuristic cost will be selected. In other word, the one that can effectively extract laughter from the viewer will be selected. This positively encourages the viewer to interact, in interactive comedy usually leading to violation of Preconditions or Executability Conditions of particular operators, more with the system. A virtuous cycle is thus established.

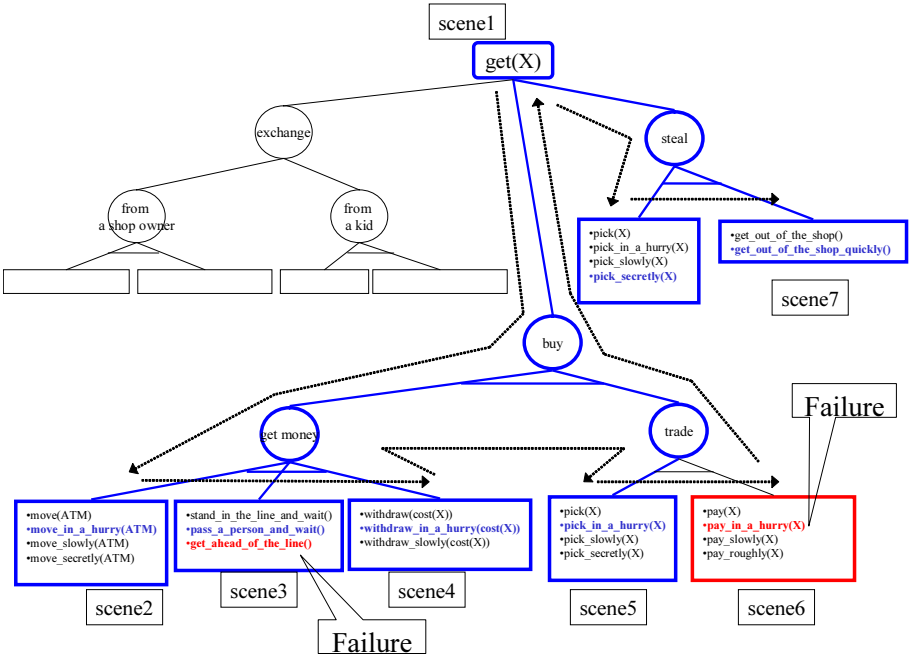


Fig. 5. The resulting plan for task **get(teddy_bear)** in the story

When an operator with unsatisfied Executability Conditions is selected, the total cost of the corresponding primitive task will be accumulated by the heuristic cost of this failed operator. If the total cost is beyond a given threshold, backtracking to the higher method in the tree will be done. Otherwise, the candidate selection is continued. If an operator with empty Executability Conditions exists in that primitive task, the operator can be selected, when necessary, to ensure the success of execution.

4 Results

Our interactive comedy system is under development. Below we show a typical planning result of the extended HTN and the corresponding screen shots.

At present, our system is aimed at playing a short visual comedy similar to the popular Mr. Bean^{TM} series. Our story starts with a scene in which the main character (henceforth called Mr. B) sees a teddy bear (notated by teddy_bear) that he has been desperately looking for. He wants to get it. Hence the goal of planning is to get the teddy bear. Using **get(X)** in Fig. 4, this task can thus be represented by **get(teddy_bear)**. For this story, the resulting plan is shown in Fig. 5. The corresponding scenes are shown in Fig. 6, the description of which is given as follows:

Scene 1: First the method **buy** is selected, then the subsequent **get money**.

Scene 2: As a candidate to be executed first, **move_in_a_hurry(ATM)** is selected. Accordingly, Mr. B moves in a hurry to an ATM.

Scene 3: When Mr. B reaches the ATM, the planning shifts to the next primitive task where **get_ahead_of_the_line()** is intentionally selected to generate a failure. This operator fails because **get_ahead_of_the_line()** has a condition at Executability Conditions, **not line_formed**, that checks whether or not a waiting line exists. Since there are two other persons already in the line, this condition does not hold. Because **get_ahead_of_the_line()** has failed, another operator, in the same primitive task, **pass_a_person_and_wait()** is then selected. As a result, Mr. B struggles to pass the person in front of him, and after succeeding to do so, waits for his turn.

Scene 4: The planning then shifts to the next primitive task where **withdraw_in_a_hurry(cost(teddy_bear))** is selected. This makes Mr. B able to acquire the necessary money. The planning for the method **get money** thus completes.

Scene 5: The planning shifts to the method **trade** where **pick_in_a_hurry(teddy_ bear)** is selected, by which Mr. B in a hurry goes to the shop and picks the desired teddy bear into his hands.

Scene 6: The planning then shifts to the next primitive task where **pay_in_a _hurry(teddy_bear)** is selected. The viewer now interacts with the system at this point. He or she teases Mr. B by taking the money away from him. This interaction results in violating **money(cost(teddy_bear))** given at Preconditions of this primitive task (cf. Fig. 3). Consequently, the method **trade** and hence the method **buy** fail.

Scene 7: Backtracking is performed which shifts the planning to the method **steal**. Under this method, **pick_secretly(teddy_bear)** is selected. So, Mr. B secretly picks the teddy bear into his hands [1]. The planning then shifts to the next primitive task where **get_out_of_the_shop_quickly()** is selected. Mr. B thus gets out of the shop quickly in order not to be caught by the shop owner.

5 Conclusions

We have described a method for planning of a character agent in an interactive comedy system. The concept of executability conditions and that of multiple candidates in a primitive task have been introduced to HTN. These concepts enable higher interactions with the viewer and more story variations. In addition, they allow for extraction of laughter with controlled failures of the character agent of interest. The extended HTN works well for a short comedy story such as the one discussed in the paper.

[1] Steal is obviously illegal. It is used here, however, to generate a humorous situation in comedy.

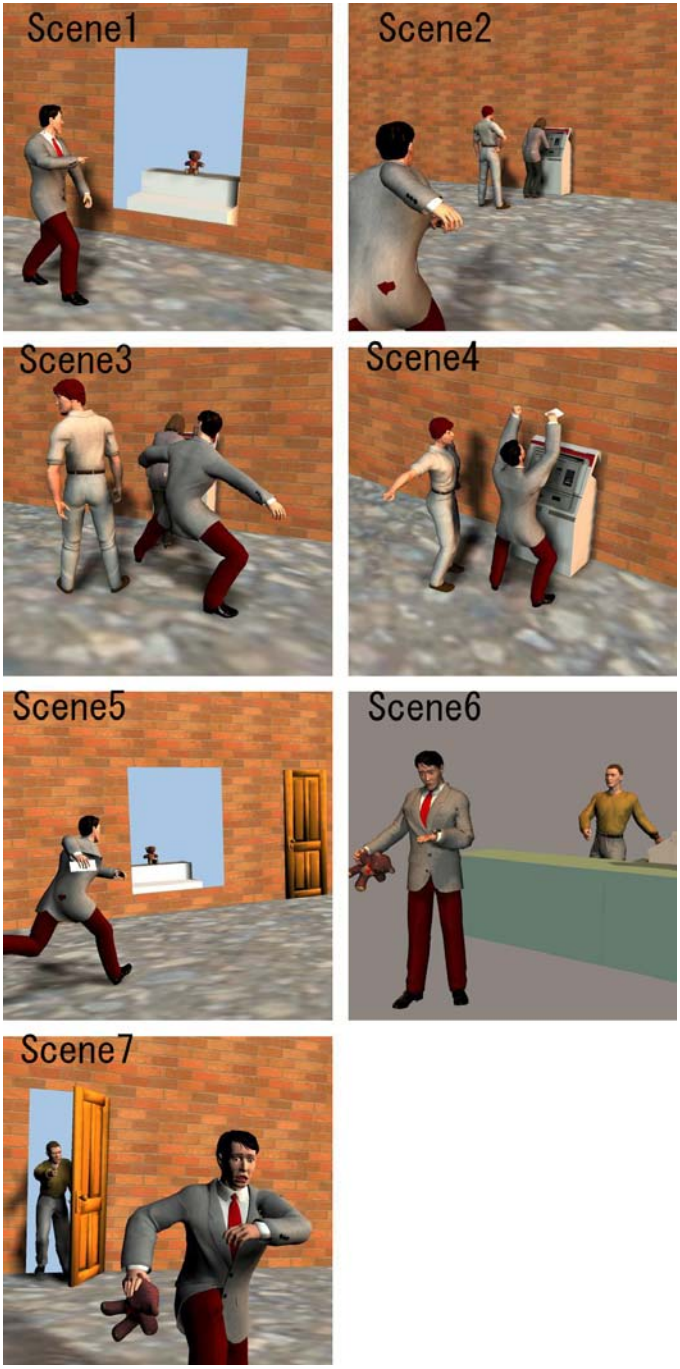


Fig. 6. The corresponding scenes of the story

In general, decomposition of a long task or a long story into subtasks in traditional HTN is difficult because one needs to ensure that the execution order of them is arbitrary. Our extended HTN also bears the same problem. As our future work, we plan to solve this and to complete as well as evaluate the system with a set of viewers.

Acknowledgements. This work has been supported in part by the Ritsumeikan University's **Kyoto Art and Entertainment Innovation Research**, a project of the 21^{st} Century Center of Excellence Program funded by the Japan Society for Promotion of Science.

References

1. Mateas, M. and Stern A.: A Behavior Language for Story-Based Believable Agents. IEEE Intelligent Systems, pp. 39–47, July–August, 2002.
2. Charles, F., Cavazza, M., and Mead, S.J.: Generating Dynamic Storylines Through Characters' Interactions. International Journal on Intelligent Games and Simulation, vol. 1, no. 1, pp.. 5–11, March, 2002.
3. Tosa, N. and Nakatsu, R.: Interactive Comedy : Laughter as the Next Intelligence System. Proc. International Association of Science and Technology for Development: Artificial and Computational Intelligence, pp. 402–405, September, 2002.
4. Cavazza, M., Charles, F., and Mead, S.J.: Generation of Humorous Situations in Cartoons through Plan-based Formalisations. CHI-2003 Workshop: Humor Modeling in the Interface, April, 2003.
5. Nau, D.S., Smith, S.J.J., and Erol, K.: Control Strategies in HTN Planning: Theory versus Practice. AAAI-98/IAAI-98 Proceedings, pp. 1127–1133, 1998.
6. Bonet, B. and Geffner, H.: Planning as Heuristic Search. Artificial Intelligence: Special Issue on Heuristic Search, vol. 129, no. 1, pp. 5–33, 2001.
7. Charles, F., Lozano, M., Mead, S.J., Bisquerra, A.F., and Cavazza, M.: Planning Formalisms and Authoring in Interactive Storytelling. Proc. of the First International Conference on Technologies for Interactive Digital Storytelling and Entertainment, Darmstadt, Germany. pp. 24–26, March, 2003.
8. Lozano, M., Mead, S.J., Cavazza, M., and Charles, F.: Search-based Planning: A Method for Character Behaviour. GameOn 2002, London, UK.
9. Shim, Y. and Kim, M.: Automatic Short Story Generator Based on Autonomous Agents. PRIMA 2002, LNAI 2413, pp. 151–162, 2002.

Author Index