COMMITMENT-BASED ANALYSIS OF ORGANIZATIONS: DEALING WITH INCONSISTENCIES

by

Shameem Shah Nawaz

B.S., Management Information Systems, Boğaziçi University, 2003

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Computer Engineering

Boğaziçi University

2007

ACKNOWLEDGEMENTS

It is my great pleasure and honor to express my heartfelt gratitude to those who have made this thesis possible.

First of all, I would like to thank my thesis supervisor, Dr. Pınar Yolum, for believing in me and for her support all along the way. The motivation, advice, guidance and enlightenment she has provided is priceless. Without her resourceful and enthusiastic assistance, I would not have been able to finish this thesis. It is truly a wonderful experience to have known and worked with a person of her caliber.

I would like to especially thank Prof. Dr. A C Cem Say and Prof. Dr. Yaman Barlas for kindly agreeing to be my committee members.

I would like to take this opportunity to state that I am ever grateful to Dr. C Say, Dr. F Gürgen and his wife Dr. N Gürgen, Dr. B Badur, A Salah, M Gönen and late Dr. T Ulus (may his soul always rest in peace) for their help during my masters study.

I am eternally indebted to my parents and my brother for their unqualified love and support throughout my life. I am grateful to Nuray Kara for her understanding, endless patience, moral-boosting support and encouragements during the thesis-writing period. Finally, I would like to express my sincere gratitude to those all who have taught me something and anything at all throughout my life, formally or informally. This thesis is dedicated to all of them.

This research has been partially supported by Boğaziçi University Research Fund under grant BAP06A103 and The Scientific and Technological Research Council of Turkey by a CAREER Award under grant 105E073.

ABSTRACT

COMMITMENT-BASED ANALYSIS OF ORGANIZATIONS: DEALING WITH INCONSISTENCIES

Multiagent organizations are composed of interacting agents. These agents are usually assigned with roles and have clearly defined tasks so that organizational goals are effectively materialized. Formal specifications of multiagent organizations allow organization designers to analyze existing organizations and reason about possible changes in the organizations. Systematic analysis of organizations can help identify potential errors in the organization early on. In this thesis we study a commitment-based approach for specifying organizations and then detecting and resolving inconsistencies and conflicts in the specifications. Additionally, we have developed a software tool to help organization designers with creation and manipulation of organizational specifications. The tool can check the workings of an organization for inconsistencies and signal the possibility or the certainty of a conflict during execution and can provide a set of suggestions for resolving conflicts. Furthermore, the tool can semi-automate the task of combining two organizational work-flows, by aggregating related properties of the organizations; and can present a higher level view of organizations. We illustrate these properties using a case study that deals with two organizations.

ÖZET

ÖRGÜTLERİN TAAHHÜT TABANLI ANALİZİ: TUTARSIZLIKLARLA BAŞA ÇIKMAK İÇİN BİR YÖNTEM

Çoketmenli örgütler birbirleriyle iletişim halindeki etmenlerden oluşur. Belirli rolleri ve görevleri olan bu etmenlerin amacı örgütün hedeflerini etkin bir şekilde gerçekleştirmektir. Örgütlerin biçimsel betimi, örgüt tasarımcısının var olan örgütleri incelemesini ve örgütlerde olası bir değişiklik hakkında fikir üretmesini sağlar. Örgütleri düzenli bir biçimde analiz etmek o örgütlerle ilgili oluşabilecek hataları erkenden teşhis etmeye yardımcı olur. Bu tez, örgütleri taahhütlere dayalı bir yaklaşımla temsil etmeye ve oluşabilecek hataları ve çelişkileri ortaya çıkartıp çözmeye yarayacak metodlar geliştirmektedir. Buna ek olarak, tasarımcıların örgütleri kolay tasarlamasını ya da gereken değişiklikleri yapmasını sağlayan bir yazılım geliştirilmiştir. Bu yazılım bir örgütün işleyişinin tutarsızlıkları kontrol ederek oluşabilecek hataları işaret edebilir ve o hataların düzeltilmesini sağlayan öneriler sunabilir. Ayrıca, birden fazla örgütün birarada çalışması için o örgütlerin benzer niteliklerini bir araya getirip, örgütlerle ilgili üst seviyeli bir görünümü kısmen otomatik bir şekilde ortaya çıkarabilir. Bu özellikler, iki örgütten oluşan bir vaka araşırmasıyla gösterilmiştir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS/ABBREVIATIONS

C_i	ith Commitment
D	*Discharge* Node in the Commitment Graph
E	Existential Time Quantifier
RC	*Release* or *Cancel* Node in the Commitment Graph
t_b	Start Point of Time
t_e	End Point of Time
t_i	ith Time Point between t_b and t_e
U	Universal Time Quantifier
ULC_i	ith Upper Level Commitment

CP	Conflicting Commitment Pair
CPT	Conflicting Commitment Pair Given Time Bounds
MAS	Multiagent Systems
TBC	Time Bound Commitment
TI	Time Interval
TP	Transition Point
ULC	Upper Level Commitment
XML	eXtensible Markup Language

1. INTRODUCTION

Multiagent organizations consist of interacting agents that fulfill a set of tasks by carrying out processes. According to Russell and Norvig [1], an agent is anything that can perceive its environment through its sensors and then act upon that environment through actuators. Examples of agents include humans, robots, or software agents. An agent is a rational agent if it can reason about possible actions and select the one that optimizes an appropriate performance measure.

Agents are seldom stand-alone systems. In many situations they coexist and interact with other agents, for instance, soccer playing robots or software agents on the Internet. Such a system, consisting of interacting agents, is called a multiagent system (MAS). Recent trends in MAS are toward the explicit design and use of organizations, which allows agents to work together within well defined roles to achieve individual and system level goals [2]. A multiagent organization is composed of agents and defines and assigns roles to agents in order to accomplish its goals. In this sense, a multiagent organization can resemble or mimic, at least theoretically, the work-flows of any real life organizations where roles and tasks are explicitly defined among members of an organization.

In multiagent organizations, it is important to specify and monitor interactions within the organization correctly, since erroneous specifications of interactions can lead to catastrophic consequences. Entire system can loose its functionality by faulty or incomplete interactions. So it is important that design flaws are detected and eventually corrected in time while designing any protocol that will be used in the organization [3]. For example, if carefully designed, problems in the flow of information in the process can be detected before the organization starts working. Importantly, the specification of organizations should be modular and flexible so that changes in an organization can be addressed immediately by making few changes. For example, if two organizational work-flows need to be combined, their specifications should be (semi-)automatically composed, rather than being created from scratch.

Protocols specify which interactions are permitted among communicating autonomous agents by necessarily constraining their behavior. Interactions among agents must be modeled in such a way that the following key aspects; autonomy, heterogeneity, opportunities and exceptions; are taken care of in order to respect the open and dynamic nature of interactions [4]. Autonomy refers to agents' ability in deciding who they want to interact with, what task they want to perform and how they want to perform. Autonomy is important for creating an effective open environment. All of the agents in a system may not be similar as they would be designed and developed by different designers adhering to diverse design principles. Protocols must facilitate meaningful and productive interactions among heterogeneous agents. Protocols should be designed in such a way that enables agents to take advantage of any opportunities that may exist in a work-flow to improve or simplify their actions. Finally, protocols should allow agents to modify their behavior to handle an exception that may occur in the system so that the ultimate goal of the organization can be achieved. Due importance should be given to these four aspects while designing multiagent protocols as they regulate interactions among agents such that the functionalities of a multiagent organization are effectively realized.

To realize successful, flexible organizations, it is necessary to enable agents to communicate with each other correctly and flexibly. This means that agents should be accommodated with meaningful constructs to interact with. In recent years, commitments have been used as a useful abstraction to represent the interactions between agents in a multiagent system. Successful and meaningful interactions among agents can be based on reciprocal commitments. Commitments refer to the obligations of one agent to another and can be manipulated easily [4]. The specification of interactions in terms of commitments allows agents to reason about their actions, which enables them to deal with unexpected situations that may arise at run time [5].

Like other protocols the commitment protocols must be unambiguous. Given that the agents in a MAS are autonomous, heterogeneous and distributed; we need to deal with possible conflicts and inconsistencies. The precise sequence of events cannot always be predicted as the number of possible orderings of events can be overwhelming.

Any attempt to analyze the commitment protocols by simple manual case analysis would be naïve. Hence an automated approach is required.

Inspired by previous work on commitments and multiagent organizations, we propose to represent the interactions in multiagent organizations as commitments, where agents interact and change the environment by making and updating their commitments as appropriate [6]. Using a formal description of commitments and incorporating these properties in a validation software tool will enable organization developers to design organizations that are logical and consistent in accordance with the system requirements [5]. Accordingly, we develop an approach that enables an organization designer to develop an organization specification, check it for conflicts, generate different views of the organization and possibly combine the work-flow of an organization with that of a second organization. We implement these steps in a software tool.

To our best of knowledge, any such automated tool is yet to be developed. Although Fornara and Colombetti developed a method for agent communication by using standardized meanings of messages and update rules within an object-oriented paradigm, they do not provide design requirements on correctness or consistency [7]. Similarly, a framework, developed by Artikis *et al.* to specify and animate computational societies, does not provide any design rules to establish the correctness of the executed societies [8].

The developed software tool should be able to check the validity, correctness and consistency of a commitment protocol where a commitment can be either fulfilled or violated or canceled. Apart from these, like any other protocols, detecting and resolving or preventing possible conflicts and inconsistencies is an important issue for commitment protocols. Resolving or negotiating possible conflicts is facilitated by the software tool in a manner such that the overall soundness of protocol is not compromised.

A commitment protocol may convey a lot more insight about a multiagent organization than is usually present in individual commitments that form a commitment protocol. A commitment protocol may give us valuable information about the agents

that are part of the multiagent organization, the roles that each agent plays in an organization, (*hierarchical*) relationships among these agents, business processes or work-flows that take place in the organization, the requirements for fulfilling a task, and so on. While analyzing and manipulating a commitment protocol, we should not only be able to extract these information but also use them effectively and flexibly for developing efficient multiagent organizations.

Contributions: This thesis studies an approach to aid organization developers to design appropriate multiagent organizations. In this respect, we make the following contributions:

- As interactions among agents are modeled in terms of commitments, there remains some scope for conflicts among commitments in a commitment protocol. We devise a way to detect possible conflicts before the protocol is put to execution.
- We propose some strategies to resolve conflicts so that the protocol designer can design consistent and conflict-free protocols.
- We present a higher level view of organizations which helps us in expressing organizations concisely without concerning about low level details.
- We devise a way to combine multiple work-flows in different organizations that can reflect real life scenarios when two organizational work-flows need to combine.
- We implement our theoretical contributions in a software tool.

Thesis Organization: The rest of the thesis is organized as follows. Chapter 2 provides a discussion about commitments, operations on commitments and existing algorithms for detecting inconsistency and conflicts among commitments. Chapter 3 discusses how the notion of time can be associated with commitments. Chapter 4 explains conflict detection among commitments. Chapter 5 provides a discussion about resolving conflicts. Chapter 6 provides a higher level abstraction of commitments. Chapter 7 discusses the architecture of the software tool we have developed. Chapter 8 provides a case study where conflicts among commitments as well as unrealistic commitments exist. Finally, Chapter 9 discusses some related works in this field and gives pointers for future directions.

2. TECHNICAL BACKGROUND

In this chapter, we explain *commitments* and *operations* on commitments in details. Commitments are at the focal point of our thesis. Or to put it in a different way, our work is built on commitments among agents. Agent interaction can be formally specified in terms of commitments. Using a set of commitments, known as *commitment protocol*, we can express the work-flow of multiagent organizations consistently and correctly. While constructing a commitment protocol, we show how to check for protocol validity and consistency during compile time so that the organization works without any glitch during run time.

We begin by defining commitment and proposition.

2.1. Commitment and Operations

A *commitment* [9], also known as base level commitment, is an obligation or a promise that can be expressed as a predicate as follows:

$$Commitment_{Id}(debtor, creditor, proposition) \ .$$

where *debtor* is an agent that makes the commitment to an agent called *creditor* to satisfy the *proposition*. An example commitment would be $C_i(x, y, p)$ where x promises y to bring about p.

Every commitment has a unique identifier, called Commitment ID. This unique ID helps to distinguish a particular commitment from other commitments that have the same debtor, the same creditor and the same proposition. If agent x promises to agent y to $Pay\ USD$ 100 twice, then a single payment of USD 100 would not be sufficient.

A number of *operations* can be applied on commitments [10]. These are:

Create operation creates or instantiates a new commitments. A commitment can only be created by its debtor. An agent, x who owes money to another agent, y can create the following commitment: $C_i(x, y, repay)$, where x promises y to repay the money owed.

Cancel operation removes a commitment. A commitment can only be canceled by its debtor. Normally, an alternative commitment is created in compensation when an existing commitment is canceled. In the previous commitment, only agent x can cancel commitment C_i.

Release operation releases the debtor of a commitment. Only the creditor of a commitment can release its debtor, and in doing so, eliminates the commitment. The creditor, y of commitment C_i can release agent x from its obligation.

Assign operation transfers a commitment to a new creditor. A commitment can be assigned to a new creditor by its current creditor. The creditor or agent y in commitment C_i can assign the commitment to a new creditor, say agent z and the modified commitment will look like the following: $C_i(x, z, repay)$.

Delegate operation transfers a commitment to a new debtor. A commitment can be delegated to a new debtor by its current debtor. The debtor or agent x in commitment C_i can delegate the commitment to a new debtor, say agent u and the modified commitment will look like the following: $C_i(u, y, repay)$.

Discharge operation satisfies a commitment. A commitment is discharged when its debtor brings about the proposition of the commitment or in other words fulfills the commitment.

2.2. Proposition

A *proposition* is the content of an assertion. It is a statement that is either true or false but not both. The statement "*The door is closed*" is a proposition. If the door in question is actually closed then the above mentioned statement will bear the truth value *true*; otherwise the truth value of this statement will be *false*.

In our work we distinguish all propositions into two broad categories: cumulative or repetitive. A *cumulative* proposition can be added with another proposition of same type to form an aggregated proposition. For instance, two cumulative propositions of "*USD 100 paid*" actually means "*USD 200 paid*". On the contrary, a *repetitive* proposition when added with another proposition of same type does not have any real effect. For instance, two propositions of "*The door is closed*" when combined together will still state the same proposition, "*The door is closed*".

2.3. Defining Work-flows of Organizations by Using Commitments

An organization is a formal group of people with one or more shared goals. An organization can be *process-related*, *functional* or *institutional*. With the spectacular advancement in information technologies, today we have many virtual organizations (one being *Wiki*), some of which are based on or inspired by real life organizations.

In a virtual World (such as *Second Life* – one of many virtual worlds that have been inspired by the science fiction novel Snow Crash by Neal Stephenson) "organization" is understood as planned, coordinated and purposeful action of human beings and computer AIs in order to construct and/or compile a common intangible product or service to its community. Just as "an organization in sociology" this action is usually framed by formal membership and form (institutional rules).

Multiagent organizations are made up of agents arranged in organizational hierarchy who continuously interact with each other to bring about a necessary or predefined task. These basic interactions, defined by institutional rules, can be expressed in terms of commitments. A commitment protocol can be devised by enlisting a set of related commitments. This set of commitments, organized in a commitment protocol depicts the work-flow of an organization. We can use this commitment protocol to check whether consistent and meaningful interactions are performed among agents.

We will elaborate how the interactions among agents in an organization can be expressed by using commitments in later chapters. For the time being, let us describe

a very simplistic organization in terms of commitments. This organization produces and sells toys. An agent, *producer* produces the toys while another agent, *seller* sells these toys. Finally, a third agent, *head* is the head of this organization. The *producer* commits to the *head* to produce toys. The formal commitment (with an arbitrary commitment ID, C_1) for this would be $C_1(producer, head, produce)$. In the same way the seller's commitment about selling toys would be $C_2(seller, head, sell)$. With these two commitments we have expressed the work-flow of this organization.

The current process of expressing interactions in terms commitments lacks the vital aspect of *time*. These commitments do not convey any information about the time period in which these commitments are expected to be valid. A major part of our work has been introducing the notion of time with commitments so that an organization work-flow is expressed more realistically.

2.4. Predicate Logic/Calculus

A *predicate* is a feature of language which can be used to assign attribute (property) to an individual thing (object). In other words, predicates are used to describe certain properties or relationships between individuals or objects [11].

Predicates are sometimes known as *propositional functions*. A predicate may be thought of as a kind of function that applies to individuals and yields a preposition. In predicate calculus, each predicate is given a name, which is followed by the list of *arguments*. The list of arguments is enclosed in parentheses. The number of elements in the predicate argument list is called the *arity* of the predicate.

For example, if we say "This book *is red*", then we have assigned the attribute redness (color being red) to this book. The phrase "*is_red*" is a predicate:

is_red(this book)

Here we have used predicate "*is_red*" to specify that this book is red. The arity of *is_red*(this book) is one.

Predicate calculus uses *quantifiers* to specify if a statement is always true, if it is sometimes true or if it is never true. Apart from containing *terms*, *predicates* and *quantifiers*, predicate calculus contains all the components of propositional calculus, including propositional variables and constants.

A classic example of what can be done with the predicate logic is the inference from the premises:

- All men are mortal.
- Socrates is a man.

to the conclusion

- Socrates is mortal.

Predicate calculus allows us to precisely specify and work on commitments.

2.5. Existing Algorithms to Detect Conflicts

A pair of commitments can be in conflict based on their proposition.

Conflicting Commitment Pair (CP) is a pair of commitments that contains conflicting propositions when debtors and creditors are identical. Commitments $C_1(x, y, p)$ and $C_2(x, y, \neg p)$ form a CP if operations on C_1 and C_2 includes only *Discharge* but not *Release* and *Cancel*. As, x cannot fulfill both p and $\neg p$, this pair of commitments is said to be in conflict. As an example, let me commit to my school to "*pay my tuition fees*". At the same time if I commit to my school *not* to "*pay my tuition fees*", my commitments to my school will be in conflict as only one of the two commitments can be brought about.

If I pay my tuition fees then there will be no way for me to discharge my second commitment or vice-versa. Hence these two commitments are in conflict.

The software tool that we have designed to detect conflict would be fed with a description of a protocol that is to be verified. Initially, a commitment graph, G, with a set of nodes (base level commitments) V and a set of edges (operations on these base level commitments) E, where $G = (V, E)$ is created [5]. A directed edge from node u to node v refers to an operation on the commitment at node u which yields node v. Example operations on commitments are: *Assign*, *Delegate*, *Discharge*, *Cancel*, *Release*, etc.

Two special nodes are added to the commitment graph initially: node D for *Discharge* operation and node RC for *Release* and *Cancel* operations. The algorithm for building the commitment graph [5] is given in Table 2.1.

Iteratively, a new node is added to the commitment graph for each commitment over the set of all commitments that can be created by the protocols. It is assumed that the graph contains a standard adjacency graph that is used to determine if a node has a neighbor. If a commitment node has at least one outgoing edge, then the commitment is said to have a neighbor.

Next, a search like mechanism is used to check if all of the commitments in the commitment graph can either be resolved or at least transferred to another node which can be resolved. A node is said to be resolvable if 1) it has at least an edge to either of the nodes RC and D; or 2) it has an edge to another node which is itself resolvable. If a commitment contains *Discharge* operation then the node corresponding to this commitment in the graph will have an edge to node D. Similarly, if a commitment contains either *Release* or *Cancel* operations then the node corresponding to this commitment in the graph will have an edge to node RC. The algorithm, given in Table 2.2, tries to visit all the commitments in the commitment graph.

Table 2.1. Algorithm for building commitment graph.

Algorithm 1 Build-commitment-graph

Parameters: CS (Set of base-level commitments),

O (Set of operations on base-level commitments)

```
1:  Create a node RC
2:  Create a node D
3:  possible-commitments = CS
4:  while (possible-commitments != nil) do
5:     Remove a commitment c
6:     Add a new node c to V
7:     for i = 1 to |O(c)| do
8:       if (O(c)[i] == delegate) then
9:          Add a new node c.delegate to V
10:         Add (c, c.delegate) to E
11:         Add c.delegate to possible-commitments
12:      else if (O(c)[i] == assign) then
13:         Add a new node c.assign to V
14:         Add (c, c.assign) to E
15:         Add c.assign to possible-commitments
16:      else if (O(c)[i] == release) || (O(c)[i] == cancel) then
17:         Add (c, c.RC) to E
18:      else if (O(c)[i] == discharge) then
19:         Add (c, D) to E
20:      end if
21:    end for
22: end while
```

Table 2.2. Algorithm for coloring graph.

Algorithm 2 Color-graph	
Parameter: G (Commitment Graph)	
1:	$visited$ = **nil**
2:	$whiteList$ = **nil**
3:	$blackList$ = **nil**
4:	**for** $i = 1$ to $\|V\|$ do
5:	**if** (V(i) $\notin$ $visited$) **then**
6:	visit(V(i))
7:	**end if**
8:	**end for**

Algorithm 2 takes a commitment graph as its input and visits every node in order to color each node with the help of Algorithm 3. In Algorithm 3, a node is marked with color *white* if the node is resolvable; otherwise it is marked with color *black*. Algorithm 3 is given in Table 2.3.

Algorithms 2 and 3 together computes a set of unresolvable commitments. The protocol designer can modify the protocol until the *blackList* computed by this algorithm is empty. The protocol consistency can also be checked by comparing all the commitments against each other to see if they have conflicting propositions. Algorithm 4 that checks protocol consistency is given in Table 2.4.

This algorithm detects inconsistencies among commitments that are not time bound. Let $C_1(x, y, p)$ and $C_2(x, y, \neg p)$ be a pair of commitments with conflicting proposition. Algorithm 4 first checks if either of the commitments have an edge to node *rc*. If none of these commitments has an edge to node *rc*, then it violates the theorem [5] that states that in an effectively progressive commitment protocol, for a pair of commitments with conflicting proposition, if either *Cancel* or *Release* operation can be performed on either of the commitment, then the commitment protocol is consistent.

Table 2.3. Algorithm for visiting nodes.

Algorithm 3 visit

Parameter: u (node)

```
1:  Add u to visited
2:  if (u.adjacentTo(D OR RC)) then
3:      Add u to whiteList
4:  else if (u.hasNeighbors()) then
5:      while (u ∉ whiteList) AND (∃ E(u, v) : v ∉ visited) do
6:          if (v ∉ visited) then
7:              visit(v)
8:          end if
9:          if (v ∈ whiteList) then
10:             Add u to whiteList
11:         else
12:             Add u to blackList
13:         end if
14:     end while
15: else
16:     Add u to blackList
17: end if
```

Table 2.4. Algorithm for checking consistency.

Algorithm 4 Check-consistency

Parameter: G (Commitment Graph)

```
1:  inconsistentList = nil
2:  for i = 1 to |V| − 1 do
3:      for j = i + 1 to |V| do
4:          Determine if V(i) and V(j) are conflicting
5:          if conflicting(V(i) and V(j)) then
6:              if NOT(∃ E(V(i), RC)) AND NOT(∃ E(V(j), RC)) then
7:                  Add V(i) and V(j) to inconsistentList
8:              end if
9:          end if
10:     end for
11: end for
```

Given a set of commitments that can be created in an organization, these existing algorithms check whether it is possible to end up with a pair of conflicting commitments, such that neither of the commitments can be canceled or released. This will create a conflict since then both commitments will need to be discharged, yielding contradictory information (i.e., p and $\neg p$). While this is useful, these algorithms do not consider time bounds on commitments.

So far we have dealt with commitments that are not bound with time. In the next chapter we introduce the notion of time and bind commitments with time intervals. When commitments are bound with time, current algorithms that we have described so far need to be modified so that we can deal with time bound commitments. We discuss this in detail in the subsequent chapters.

3. ADDITION OF TIME

Commitments are generally considered without any notion of time frame. But a commitment should not vanish instantly. Nor should it persist for ever. Commitments used without any realistic time frame cannot capture real life subtleties. Hence it is only logical to introduce the notion of time with commitments. A commitment should be clearly bound by a time frame (i.e. an interval) in which a commitment exists.

We follow a similar, but simpler time model of [12]. Our time model is based on a collection of discrete time points that are linearly ordered. This approach enables us to deal with realistic scenarios when we try to specify organizational interactions of agents in terms of commitments. It also helps us in dealing with avoiding subtle ambiguities that can arise in real life situations and resolving conflicts among commitments.

In this chapter, we discuss how we can use the notion of time with commitments. A time bound commitment exists only in the specified time interval which is delimited by two time points. Following subsections discuss these *time*-related concepts and how we can bind commitments with a specific time interval.

3.1. Time Point

A *time point* is a specific and discrete moment in time. An action, such as creating a commitment or an event, such as fulfillment of a commitment can occur at a particular time point. For example, we can say that a commitment is created at t_x and this commitment is delegated at t_y where t_x and t_y denote two distinct time points.

There are two special predefined time points in our model: one is t_b (i.e., beginning of time in the linear ordering of time points) and the other is t_e (i.e., ending of time). Each time point defined in the commitment protocol must be a time point that is later than t_b but earlier than t_e.

Time points are linearly ordered and a binary predicate "$earlier(t_x, t_y)$" denotes that time point t_x comes earlier than time point t_y in the linear ordering.

3.2. Time Interval

A *time interval* is a time frame bounded by two time points. For example, from t_x to t_y denotes a linear time interval bounded by t_x and t_y; both inclusive. So, a time interval is delimited by two time points t_x and t_y where the time interval starts at t_x and ends at t_y. For any such interval, $t_x \leq t_y$.

In our model, a commitment can be valid only at a particular time point or it can be valid for a period of time as specified by a time interval.

3.3. Transition Point

A *transition point* [13] is a hypothetical time point which helps us specify a time point when a proposition reverses. It is a time point, t_p at which a proposition, p ceases to exist and another proposition (generally, the opposite of p) becomes valid, but these two propositions are not valid at the same time. Time point, t_p can be extended as having a time point, t_p^- that immediately precedes t_p, and another time point t_p^+ that immediately follows t_p. So, if proposition p reverses at transition point t_p, it is considered that p ceases to be valid at t_p^- and $\neg p$ (opposite/negation of p) starts to be valid at t_p^+. In this way a conflict less transition can be realized for a proposition. As an example, an agent commits to keep the door shut until t_p but recommits to keep the door open from t_p onward.

This (transition point) is different from the following scenario where both p and $\neg p$ are valid at the same time point, t_p. If p is keeping the door shut and $\neg p$ is keeping the door open, then we come to a point where we cannot know for sure whether the door will be open or closed at the time point t_p. In such a case, a conflict in propositions is a possibility.

3.4. Time Quantifier

A time interval can be quantified as either *universal* or *existential*. If the time interval of a commitment is *universal* then the proposition of that commitment has to hold at every instant in the whole interval. But, if the time interval of a commitment is *existential* then the proposition has to hold at least once in one particular time point during that time interval.

If a commitment of keeping the door open is quantified with *universal* quantifier, the door has to remain open for the entire time interval. On the other hand, if the same commitment were quantified with *existential* quantifier, keeping the door open for a fraction of the entire time interval would have sufficed.

3.5. Binding Commitments with Time

A commitment can be bound by two time points that form a time interval. "$FromTo(t_x, t_y)$" is a binary predicate that defines a time interval that is bound by from time point t_x to time point t_y given that t_y does not come earlier than t_x in the linear ordering of time points. Both t_x and t_y are included in the time bound.

Other time bounds can be originated from this basic "$FromTo(t_x, t_y)$" time bound that have analogies in nature (i.e., natural languages). In the following derivations, t_b is the beginning and t_e is the ending of time while t_x and t_y are two arbitrary time points such that $t_x \leq t_y$.

1. $At(t_x) \Rightarrow FromTo(t_x, t_x)$ This states that a proposition holds instantaneously only at the time point t_x.
2. $After(t_x) = Since(t_x) \Rightarrow FromTo(t_x, t_e)$ This, as depicted in Figure 3.1, states that a proposition holds from t_x until the end time t_e.
3. $Until(t_x) = Before(t_x) = By(t_x) \Rightarrow FromTo(t_b, t_x)$ This, as depicted in Figure 3.2, states that a proposition holds until t_x from the beginning of time t_b.

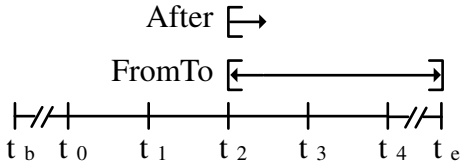

Figure 3.1. $After(t_2)$ mapped to $FromTo(t_2, t_e)$. t_e is the ending of time.

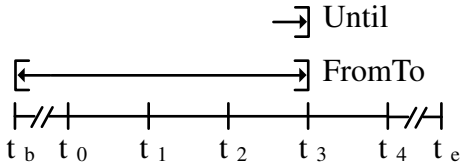

Figure 3.2. $Until(t_3)$ mapped to $FromTo(t_b, t_3)$. t_b is the beginning of time.

Every time interval, except $At(t_x)$, has a time quantifier, either *universal* or *existential*, associated with it. $At(t_x)$ is essentially *existential*. As all time bounds can be mapped to $FromTo(t_x, t_y)$, we only focus on this time bound as we look for conflicting scenarios between two time bounds. A tag, $[t_x, t_y]$ replaces the predicate $FromTo(t_x, t_y)$ which indicates a time interval bound by start time, t_x and end time, t_y for the sake of concise and formal specification of a time bound commitments.

A time bound commitment (TBC), can now be stated as

$Commitment_{Id}(debtor, creditor, proposition)$ $[time\ bound]$ $Quantifier$.

Time quantifier *universal* is denoted with the letter U while *existential* is with E. In order to elaborate time bound commitments, let us give some examples of commitments that include both types of proposition.

$C_1(x, y, p)$ [1:00pm−1:10pm] E – x commits p to y anytime between 1:00pm and 1:10pm. This is an *existential* commitment. If p is *turning the light on*, then p is a *repetitive* proposition, as once the light is turned on, committing to turn the light on again has no effect.

But if p is *paying 100 USD*, then p can be *cumulative* as committing p again can mean committing to pay *another* 100 USD.

$C_2(x, y, q)$ [4:00pm−5:00pm] $U - x$ commits q to y for the time duration between 4:00pm and 5:00pm. This is a *universal* commitment. If q is *playing tennis*, x commits to play tennis in the specified time duration. The proposition here is repetitive as committing another q for the same time duration does not have any real meaning.

But on the other hand, if q is *teaching mathematics* to a student, another q might mean *teaching mathematics* to a group of two students in the same time interval. In this sense the proposition is *cumulative*.

Binding a commitment with a time period specifies an interval in which the commitment is expected to remain valid. As such, any operation that can be applied on a time bound commitment is bounded by a time interval based on the time interval of the commitment.

Create a commitment has to be created any time before its specified time interval.

Release the debtor of a commitment can be released from its obligation by the creditor before the time interval of the commitment expires.

Cancel the debtor can cancel a commitment before it becomes valid.

Discharge a commitment can only be discharged in the specified time period that temporally binds the commitment.

Delegate a commitment can only be delegated before the start of its time interval.

Assign a commitment can only be assigned before the start of its time interval.

3.6. Conflicts in Time-bound Commitments

In the previous chapter we have defined a pair of conflicting commitments or *CP*. Now, as we have associated time with base-level commitments, we can define conflicting commitment pair that are time bound.

Conflicting Commitment Pair given Time Bounds (CPT) is a pair of commitments that contains conflicting propositions in overlapping time interval when debtors and creditors are identical. As commitments are bound by time points or time intervals,

a CP can be resolved if they are valid at different time points or in non-overlapping time intervals. For instance, if commitment $C_1(x, y, p)$ is bound at time point t_1, and commitment $C_2(x, y, \neg p)$ is bound at time point t_2, where t_1 and t_2 are two different time points, this pair of commitments is resolvable although the commitments contain conflicting proposition. But this pair of commitments will still be conflicting if they are valid at the same time point or if their time intervals overlap.

As an example, let us consider previous chapter's example again but this time binding those commitments with time intervals. The first commitment was I committing to my school to "*pay my tuition fees*". Let us associate time with this commitment by stating that I commit to my school to "*pay my tuition fees*" in the second week of September. Binding second commitment with time as well, and the time interval being the first week of September, the time bound commitment is now I committing to my school *not* to "*pay my tuition fees*" in the first week of September. Since the time intervals of these two commitments are not overlapping, these two commitments are not conflicting any longer. Both commitments can now be fulfilled in their respective time periods.

But if the time interval of the second commitment were the second week of September, they would still be conflicting given time bounds as to commit two contradictory propositions in the same time interval. So fulfilling one of these two commitment will make it impossible to fulfill the other one in the specified time period.

As we are now capable of specifying commitments with time bounds, in the next chapter we will discuss how to detect inconsistencies and conflicts among commitments that are time bound. We will see how conflicts among commitments are dependent on time bounds.

4. CONFLICT DETECTION

In the previous chapter we have defined a pair of conflicting commitments given time bounds.

In the existing algorithms that detect conflicts among commitments, time bounds on commitments are not considered. When time bounds are introduced on commitments, a pair of conflicting commitments can be rendered not conflicting, if and only if, the conflicting propositions are valid at different time points or in different time intervals.

4.1. Conflict Detection

An extension of the current algorithms would be to take a CP and check if they still remain conflicting given time bounds. In doing so, time bounds associated with the commitments of a CP are compared to check if there is any overlapping section of time between these time bounds. CPs can be rendered not conflicting only when the intersection of their time bounds is *nil*. Otherwise, they continue to remain conflicting and form CPT.

An additional algorithm, Algorithm 5, is developed which is used to determine which CP is also a CPT by iterating over the set of all CPs and comparing the time intervals of commitments in each CP. The time intervals of a pair of commitments that form a CP are compared to check if they overlap. If they overlap, it would mean that a pair of commitments with conflicting propositions need to be valid at the same time.

The set of CPTs is intuitively a subset of CPs. A CP that is not in the set of CPTs can be rendered as not conflicting given time bounds. A pair of commitments containing conflicting propositions can be valid if their time intervals are not overlapping. Algorithm 5 is as follows:

Table 4.1. Algorithm for detecting CPT.

Algorithm 5 Detect-CPT

Parameter: SCP (Set of CPs)

```
1:  conflictGivenTimeList = nil
2:  noConflictGivenTimeList = nil
3:  for i = 1 to |SCP| do
4:      Compare time intervals of SCP(i)
5:      if (time intervals overlapping) then
6:          Add SCP(i) to conflictGivenTimeList
7:      else
8:          Add SCP(i) to noConflictGivenTimeList
9:      end if
10: end for
```

If a base level commitment is in conflict, any commitment derived from this commitment (by using either *delegate* or *assign* operation) will also be in conflict. The following two base level commitments are in conflict due to conflicting propositions:

$C_a(y, x, q)\ [t_u - t_v]\ U$

$C_b(y, x, \neg q)\ [t_u - t_v]\ U$

Now if y in commitment C_b decides to delegate to z, the derived commitment $C_c(z, x, \neg q)\ [t_u - t_v]\ U$ will inherit the conflict with C_a. In general, conflicts can also arise when debtors or creditors differ but the commitments contain conflicting propositions.

As an example let us consider the following scenario. Commitment C_a is where y commits to "*keep the main door open*" for x during the time interval from 8:00am to 8:30am and Commitment C_b is where y commits *not* to "*keep the main door open*" (or in other words, "*keep the main door closed*") for x during the time interval from 8:00am to 8:30am. These two commitments are clearly in conflict during the specified time interval.

Now y decides to delegate its second commitment C_c of "*not keeping the main door open*" to z, and creating a new commitment in the process. The derived commitment C_c will be in conflict with commitment C_a as if either of these commitment is fulfilled, the other one will not be fulfilled. There is no way of "*keeping the door both open and closed*" at the same time. As a result, these two commitments C_a and C_c will be in conflict although their debtors are not the same.

Conflicts among commitments can arise in different scenarios when their time bounds and time quantifiers differ.

4.2. Conflict Scenarios

Since all other time bounds can be mapped to $FromTo(t_x, t_y)$ it is sufficient to show conflicts and no-conflicts between two $FromTo(t_x, t_y)$ time intervals. Let $C_1(x, y, p)$ and $C_2(x, y, \neg p)$ be a pair of commitments with conflicting proposition, hence a CP. These commitments are bound by $FromTo(t_x, t_y)$ time intervals.

Let t_0, t_1, t_2, t_3, and t_4 be five different time points and $t_i < t_k$ *iff* $i < k$.

4.2.1. Non-overlapping Time Intervals

Let, C_1 be bound by $FromTo(t_1, t_2)$ and C_2 be bound by $FromTo(t_3, t_4)$. Since the time intervals are not overlapping (See Figure 4.1) there is no conflict given time bounds.

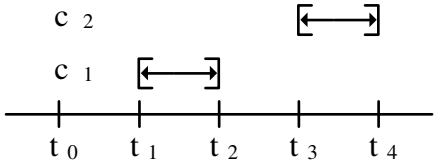

Figure 4.1. C_1 and C_2 will not be in conflict as their respective time bounds $FromTo(t_1, t_2)$ and $FromTo(t_3, t_4)$ do not overlap.

The possibility of a conflict given time bound arises when time intervals associated with a CP are overlapping. A conflict can be further classified as either *possible* or

certain depending on the time quantifiers of the time intervals of a CP. While a *certain* conflict is bound to happen if no corrective measures are taken, a *possible* conflict may be avoided if the commitments of a CP are discharged at different times.

In the following sub-sections, we identify some of the key conflicts that can arise and mark whether the conflict is possible or certain.

4.2.2. Identical Time Intervals

Let, both C_1 and C_2 be bound by $FromTo(t_1, t_2)$ as shown in Figure 4.2. If both the interval of C_1 and C_2 are existential, there is a possibility of a conflict if the two commitments are discharged at the same time. For all other three variations, it is certain to have a conflict since at least one of the commitments is bound with a universal time bound. That is, if C_1 is bound with a universal quantifier, then p has to hold at all times between t_1 and t_2. At the same time, for C_2 to be discharged, $\neg p$ should hold at at least one point between t_1 and t_2, which raises a conflict.

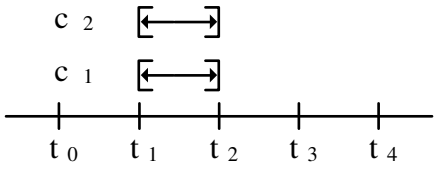

Figure 4.2. C_1 and C_2 will be in conflict in the time interval $[t_1 - t_2]$.

Table 4.2. Conflict type based on C_1 and C_2's time quantifier in the time interval $[t_1 - t_2]$.

C_1**'s Quant.**	C_2**'s Quant.**	**Conflict**
Universal	Universal	Certain
Universal	Existential	Certain
Existential	Universal	Certain
Existential	Existential	Possible

As an example of *certain* conflict between two commitments having the same time interval when one is *universally* quantified but the other is *existentially* quantified, let

us consider the following situation: a professor is scheduled to give a lecture on Mondays between 2 pm and 3 pm (Commitment 1, universal). If the same professor plans to see the department head at any time during this period (Commitment 2, existential), there will certainly be a conflict.

4.2.3. Containing Time Intervals

Let C_1 be bound by $FromTo(t_1, t_4)$ and C_2 be bound by $FromTo(t_2, t_3)$, so that the first interval contains the second interval as shown in Figure 4.3. Conflict will now occur only in the time interval $FromTo(t_2, t_3)$. If the time quantifier of the commitment with the larger time interval (C_1, in this case) is *universal*, the conflict is a certainty. On the other hand, if its time quantifier is *existential*, the possibility of a conflict remains, although it is no longer certain. Please note that the time quantifier of the other commitment does not have any effect on determining the conflict type.

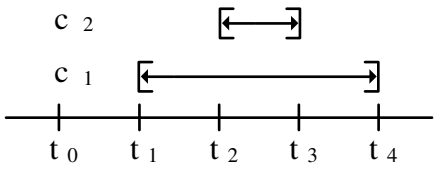

Figure 4.3. C_1 and C_2 will be in conflict in the time interval $[t_2 - t_3]$.

Table 4.3. Conflict type based on C_1 and C_2's time quantifier in the time interval $[t_2 - t_3]$.

C_1's Quant.	**C_2's Quant.**	**Conflict**
Universal	Universal	Certain
Universal	Existential	Certain
Existential	Universal	Possible
Existential	Existential	Possible

Let us now shrink the time interval of Commitment 2 such that the time bound of the modified commitment is from 2:30 pm to 2:40 pm. Although we change the time bound, it still is a subset of the original time bound of Commitment 2. Hence, the certainty of a conflict between Commitment 1 and Commitment 2 still remains.

4.2.4. Intersecting Time Intervals

Let C_1 be bound by $FromTo(t_1, t_3)$ and C_2 be bound by $FromTo(t_2, t_4)$ such that the two intervals are intersecting as shown in Figure 4.4. Conflict can occur at the time interval $FromTo(t_2, t_3)$. When both of the commitments' time quantifier is *universal*, it is certain that a conflict will occur in the intersecting time interval. For other combinations of time quantifiers, conflict is a possibility.

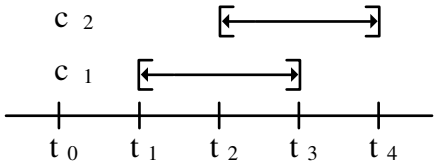

Figure 4.4. C_1 and C_2 will be in conflict in the time interval $[t_2 - t_3]$.

Table 4.4. Conflict type based on C_1 and C_2's time quantifier in the time interval $[t_2 - t_3]$.

C_1's Quant.	**C_2's Quant.**	**Conflict**
Universal	Universal	Certain
Universal	Existential	Possible
Existential	Universal	Possible
Existential	Existential	Possible

Let us now consider the situation when the professor plans to attend a conference on a Monday afternoon for the entire afternoon (Commitment 3, universal). This situation will also lead to a *certain* conflict with Commitment 1 as she will not be able to bring about both of her commitments. Let us now modify Commitment 2 such that instead of attending the conference she promises to see the department head at some time during the entire afternoon (Commitment 4, existential). In this situation given that she is scheduled to teach between 2 pm and 3 pm (Commitment 1, universal), we cannot tell for sure that there will be a conflict. All we can say is that there will be a conflict if she tries to see the department head during the lecture hour, but she can conveniently execute Commitment 4 either before the class hour or after the class hour in that afternoon. So the possibility of a conflict depends on the time when she brings about Commitment 4.

Now we know, how and in which circumstances conflicts can occur among commitments. We also know the conflict type of a possible conflict. In the following chapter we try to find ways on how to resolve conflicts among commitments.

5. RESOLVING CONFLICTS

In Chapter 4, we have seen how a pair of commitments can be in conflict with each other given time bounds. While designing a commitment protocol for an organization, a protocol designer would naturally want to have a protocol that is free from conflicts among commitments. The main objective is to have a multiagent organization that operates smoothly during run time by following a conflict-free commitment protocol. To this end, it is of vital importance to detect beforehand and then resolve conflicts during compile time so that the multiagent organization works as intended during run time without any glitch.

In this chapter we look into ways of resolving conflicts that may arise during compile time. But let us first elaborate a few key terms here.

Organization State: This refers to the propositions and commitments that hold at a particular time point for an organization.

Equivalent Commitments: This refers to a set of commitments that are not identical but have the same effect on the events. The following commitment
$C_m(y, x, \mathit{Open\ Door})\ [t_u - t_v]\ E$ – where y commits to open the door to x
is equivalent to commitment
$C_n(z, x, \mathit{Open\ Door})\ [t_u - t_v]\ E$ – where z commits to open the door to x
as both agents y and z commit to open the door (identical proposition) for agent x.

This can be specified as, $C_m \approx C_n$. Realizing only one of the commitments has the effect of realizing all the other equivalent commitments.

Types of Proposition: As stated earlier, we divide all propositions into two broad categories:

1. Repetitive: Let us consider commitments C_m and C_n, stated earlier while defining equivalent commitments, where both agents y and z promise agent x to open the door. Since opening the door only once by either of the agents would suffice for x to be able to pass through, both agents y and z need not fulfill their individual commitments independently given that at least one agent fulfills its commitment. In this respect, the proposition in these two commitments is repetitive. As soon as one of the commitments is fulfilled, the organization state changes to a new state. But fulfilling the remaining commitment does not cause the organization state to change.

   Formally, Let S_1 and S_2 be two organization states such that the organization moves from S_1 to S_2 when proposition p is applied. If the organization stays in S_2 for all other consecutive applications of p, then p is a *repetitive* proposition.
2. Cumulative: Let us now consider the following two commitments,

   $C_e(y, x,\ Pay\ USD\ 100)$ [*by tomorrow*] E – where y commits to pay USD 100 to x and

   $C_f(z, x,\ Pay\ USD\ 100)$ [*by tomorrow*] E – where z commits to pay USD 100 to x.

   Unlike in the case of *repetitive* propositions, fulfilling only one of the two commitments does not mean that the other commitment need not be fulfilled. Agent z is supposed to receive a combined payment of USD 200 from agents y and z each paying USD 100 individually. Therefore, the proposition used in the two commitments is *cumulative*.

   Formally, a proposition p is a *cumulative* proposition if it always causes the organization move from current organization state to another state whenever p is applied.

5.1. Ways of Resolving Conflicts and Anomalies in the Graph

As the commitment protocol developer's primary goal is to compile a set of consistent and conflict-free commitments, the need for dealing with anomalies in the commitment graph and conflicts in commitments, and resolving them is of paramount importance.

Anomaly in the commitment graph mainly arises when some nodes are marked black. A node is marked black if the commitment corresponding to this node cannot be resolved. A commitment cannot be resolved if 1) it does not contain at least one of the following operations in its operation set: *Discharge*, *Release*, and *Cancel*; or 2) this commitment's debtor cannot be delegated to a new debtor by *Delegate* operation nor can it be assigned to a new creditor by *Assign* operation.

Consequently, a black node does not have an edge to either RC or D node in the commitment graph. Nor does it have an edge to another resolvable node in the graph. In order to make this node resolvable, we can add either *Release* or $Cancel$ operation to the operation set of the commitment that corresponds to this node in order to connect this node with RC. Similarly, we can add $Discharge$ operation so that this node is connected to the D node in the graph. Similarly, if possible, we can add *Assign* or *Delegate* operations such that this node is connected with another resolvable node.

As a black node indicates the presence of an unresolvable commitment in the commitment graph, the commitment protocol designer can make sure that all the commitments in the commitment protocol are resolvable by doing necessary modifications on unresolvable commitments.

A commitment protocol free of any unresolvable commitments can still contain conflicting commitments. There are several ways of resolving conflicts among commitments. They are as follows:

Discarding a Commitment Getting rid of a commitment that is in a CPT is the easiest strategy of dealing with conflicts. As one of the commitments in a CPT is discarded, the other commitment automatically becomes conflict-free. But discarding a commitment may always not be desirable as it might prevent us from representing an organization in a correct way. So rather than simply discarding a commitment in a CPT, we can try to modify the commitment so that the conflict is resolved.

Modifying a Commitment We know that for a possible conflict to arise the debtors and creditors of a pair of commitments have to be the same. If possible, we can form a substitute commitment for a commitment that may cause a conflict by changing either the debtor or the creditor. Modifying a commitment in this way may constitute a fundamental change in the organization such that an agent's task is either assigned or delegated to another agent. Sometimes, it may be the case that an agent's task cannot be reassigned, making modification of a commitment impossible. In these circumstances we can try, if possible, shifting the time bounds of the commitments in a CPT.

Shifting Time Bounds One way of shifting time bounds of the commitments that are in conflict is to divide the conflicting time interval into two equal time intervals and assigning one interval exclusively to one commitment and the other time interval exclusively to the other commitment. As a result, the conflicting time interval becomes nil which in turn makes the CPT not conflicting. If a fair distribution of conflicting time interval is not possible, an unequal distribution can be done. This can be achieved by introducing constraints between the pair of commitments such that one of the commitments becomes valid and discharged. As soon as this commitment is discharged, this commitment ceases to be valid while the other commitment becomes valid.

Another way of shifting time bounds is to keep one commitment's time bound unchanged while shifting the other commitment's time time bound away from the conflicting time interval until the conflicting time interval disappears.

A commitment's proposition may be constrained on a different proposition such that in order to discharge this commitment the other proposition has to be fulfilled earlier or simultaneously. In other words, although a commitment may not be in a conflict with other commitments, the commitment itself may not be realistically discharged. For instance, a commitment of producing a particular product can only be successfully discharged if there are commitments for suppling the necessary raw materials.

In addition, duplicate commitments, if any, in the commitment protocol are detected and discarded until no duplicate commitment remains in the protocol. By duplicate commitments, we mean identical commitments as opposed to equivalent commitments.

5.2. An Example Scenario of Conflict Resolution

Following is a scenario where there seems to be a probable conflict between two commitments made by agent y to agent x. In this scenario, agent x commits to agent y to pass through a door within a specified time period while agent y promises to open a door for agent x to pass through and then close the door once agent x has passed through. Agent x cannot specify an exact time point for passing through the door but it can specify a period of time within which it can pass through the door. Due to this, agent y cannot specify beforehand when exactly to open the door and when exactly to close the door. All it can do is specify a period of time – the same period of time within which x is to pass through, for both of its commitments. These two commitments contain contradictory propositions (opening versus closing the door) within the same period of time. Hence a possibility of conflict arises between opening and closing the door regardless of x's action.

We first enlist the propositions, agents and commitments used in the scenario that we have described above.

Propositions:

p: Passing through the door.
q: Opening the door.
$\neg q$: Closing the door.

For these propositions (and commitments that contain these propositions), we do not make any assumption regarding the state of the door (whether it is already open or closed).

Agents: x and y.

Commitments:

$C_1(x, y, p)$ $[1:00pm - 1:10pm]$ E – x commits to pass through the door anytime between 1:00pm and 1:10pm.

$C_2(y, x, q)$ $[12:59pm - 1:09pm]$ E – y commits to x to open the door between 1:00pm and 1:10pm.

$C_3(y, x, \neg q)$ $[1:00pm - 1:11pm]$ E – y commits to x to close the door between 1:00pm and 1:10pm.

Commitments made by agent y, namely C_2 and C_3 possess the possibility of conflict (opening/closing the door at the same time instant) during the overlapping time period between 1:00pm and 1:10pm.

There are two alternative ways to resolve this conflict. First alternative is to rearrange and decrease the respective time bounds of commitments C_2 and C_3 so that they do not overlap. For instance, if we make a fair distribution of time between C_2 and C_3, the revised commitments, C_4 and C_5, respectively become as follows:

$C_4(y, x, q)$ $[12:59pm - \mathbf{1:04pm}]$ E

$C_5(y, x, \neg q)$ $[\mathbf{1:05pm} - 1:11pm]$ E

Although it might be logical to divide time bounds between conflicting commitments in many other scenarios, dividing time period causes commitment C_1 to be modified accordingly as agent x now needs to pass through the door before 1:05pm.

The better alternative is to create a variable transition time point, t_v. This is the moment when agent x actually passes through the door. Using this transition point

we can now rearrange commitments C_2 and C_3 in such a way that revised C_2 (or C_6) remains valid until t_v and revised C_3 (or C_7) becomes valid only after t_v as follows:

$C_6(y, x, q)\ [12:59pm - t_v]\ E$

$C_7(y, x, \neg q)\ [t_v - 1:11pm]\ E$

Here, some sort of constraints (or precedence rules) based on the temporal ordering of events may be introduced. For instance, it can be specified that $C_6 < C_7$ (i.e. C_6 must be realized in order for C_7 to be valid and for C_6 to cease to be valid, C_1 must be realized). Accordingly, if an event, e (for instance, the act of opening the door) needs to happen before a particular commitment (committing to close the door) can become valid, it can be specified as follows: $e < C_7$.

So, in resolving conflicts, any such constraints, if given can be used to resolve conflicts. Otherwise; the first alternative can be used.

Furthermore, although commitment C_1 is neither in conflict with C_2 nor with C_3, the door has to be opened by y (assuming that the door usually remains closed and only y can open the door) so that x can pass through. The proposition of C_1 is constrained on the proposition of C_2. If C_2 did not exist, C_1 would have been an unrealistic commitment as x would not be able to pass through the door if y did not commit to open the door.

6. OPERATIONS ON ORGANIZATIONS

One of the main objective of this thesis is to use commitments in order to represent the interactions among agents in a multiagent organization. To this end, we have developed necessary concepts in the previous chapters so that interactions among agents are realistically represented as TBC. Now we shift our focus on what can be done with these representations. We first explain an upper level abstraction of commitments and then we show how multiple multiagent organizational work-flows can be combined together.

6.1. Upper Level Aggregation of Commitments

In a commitment protocol some commitments may be very similar to each other. A set of commitments describe almost same interactions among agents. The difference among them is minor; such as different debtors making same commitment to one creditor or one debtor making a particular commitment multiple times spanning many different time periods.

Commitments can be combined together into an aggregate commitment or to create *upper level commitments* (ULC) based on some logical similarities among base level commitments. Creating ULCs are important as they provide us upper level views of an organization. For example, each department in a university can be working independently, but periodically we might need to view the activities of the university as a whole rather than individual departments. This requires the commitments of each part to be aggregated to yield a bigger picture of an organization. aggregation.

6.1.1. Aggregation on Time

If debtor, creditor and proposition are identical, then commitments having consecutive (contiguous) time bounds with *universal* time quantifier can be aggregated on time bounds to create an upper level commitment.

Let $C_1(x, y, p)$ $[t_1 - t_2]$ U, $C_2(x, y, p)$ $[t_2 - t_3]$ U and $C_3(x, y, p)$ $[t_3 - t_4]$ U be three base level commitments having identical debtor, creditor and proposition. $t_1 < t_2 < t_3 < t_4$ are linearly related time points. Since their time quantifiers are the same (all *universal* in this case), these three commitments can be combined together on time bounds, as depicted in Figure 6.1, to create an upper level commitment, $ULC(x, y, p)$ $[t_1 - t_4]$ U.

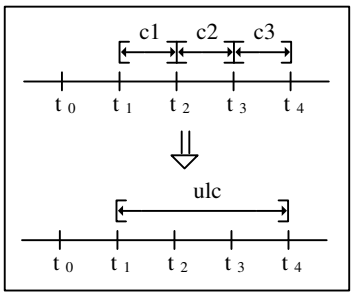

Figure 6.1. C_1, C_2, and C_3 aggregated to ULC based on time.

An example of this type of aggregation would be the commitments of a visiting professor who commits to a university to teach. The professor commits to teach during the fall semester. She makes another commitment to teach during the spring semester as well. Aggregating on time as we know that combining these two semesters would give us an academic year, the professor commits to the university to teach for a *full academic year*.

Unlike base level commitments with *universal* time quantifier, base level commitments with *existential* time quantifier cannot be aggregated in this manner as the proposition needs to hold more than once in different time intervals.

Similarly a group of commitments with a mixture of both *universal* and *existential* time quantifiers cannot be aggregated.

6.1.2. Grouping Agents

When debtors in commitments are different but all other components of commitments are same, we can group together the debtors to an upper level collection of debtors. All of the commitments should be quantified with the same time quantifier.

For example, if we have three commitments: $C_1(x_1, y, p)\ [t_1 - t_2]\ E$, $C_2(x_2, y, p)\ [t_1 - t_2]\ E$ and $C_3(x_3, y, p)\ [t_1 - t_2]\ E$ and if we know that $X = \{x_1, x_2, x_3\}$ then we can substitute these three base level commitments by an upper level commitment, $ULC(X, y, p)\ [t_1 - t_2]\ E$ as shown in Figure 6.2. The important point here is that every member of

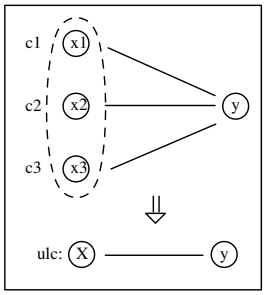

Figure 6.2. C_1, C_2, and C_3 aggregated to ULC based on agents.

the group needs to make the same commitment. If one does not make the commitment that all other group members make, we cannot create the upper level commitment on behalf of the entire group.

The problem here is about knowing group hierarchies and group compositions beforehand. The relation $X = \{x_1, x_2, x_3\}$ needs to be clearly stated (or known) before we try to formulate the upper level commitment that combines agents – debtors and creditors alike.

When every student in a class individually commits to their professor that they would study hard in order to get good grades, we can say that the whole class (collection of students) is committed to study hard.

Likewise, when a debtor commits same proposition to a group of creditors, these creditors can be combined as a group quite the same way we have combined the group of debtors as in the previous example.

A student makes individual commitments to every course instructors that he would study hard so that he can get good grades in the courses he has taken in a particular semester. The aggregation of his commitments would be him committing to a group of course instructors that he would study hard for that semester.

6.1.3. Conceptually Upper Level Commitments

A group of commitments may be conceptually related. Let us consider the following scenario where a professor plans to offer a semester-long course. This is an upper level commitment as it contains many lower (or base) level commitments:

(1) the professor commits to the department to prepare the course schedule at the beginning of a semester,
(2) she commits to the students taking this course to hold weekly lectures, and finally,
(3) she commits to the students to evaluate their performance at the end of the semester in due time.

Now let us view it bottom-up. When we are given these three (numbered 1, 2, and 3) base level commitments, we should be able to formulate the corresponding upper level commitment that the professor commits to offer a course if we agree that committing these commitments individually sum up to committing to offer a course for a semester. Here, we assume that the concept of offering a course consists of several sub-concepts, such as in our case preparing course, giving weekly lectures and evaluating students' performance.

6.1.4. Aggregation on Transitivity of Commitments

A set of transitive base level commitments may form an upper level commitment. Let us consider the following three base level commitments:

$C_1(x, u, p)\ [t_1 - t_2]\ E$,
$C_2(u, y, p)\ [t_2 - t_3]\ E$, and
$C_3(y, z, p)\ [t_3 - t_4]\ E$.

These three commitments contain an inherent transitive relation. First, x promises p to u, then u promises p to y who in turn promises p to z. The only constraint here is that for u to make commitment C_2, C_1 has to be already fulfilled, and so on.

From these three base level commitments we can formulate the upper level commitment, $ULC(x, z, p)\ [t_1 - t_4]\ E$ as shown in Figure 6.3.

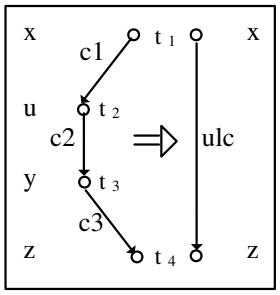

Figure 6.3. C_1, C_2, and C_3 aggregated to ULC based on transitivity of commitments.

6.1.5. Aggregation on Transitivity on Fixed Time

This type of aggregation is similar to aggregation on transitivity of commitments except that all of the commitments that forms this aggregation have same time intervals with same time quantifiers. To clarify further, we use the previous set of commitments but this time these commitments are bound with identical time intervals.

$C_1(x, u, p)\ [t_1 - t_2]\ E$,
$C_2(u, y, p)\ [t_1 - t_2]\ E$, and
$C_3(y, z, p)\ [t_1 - t_2]\ E$.

Since these commitments contain a transitive relation, a ULC is created as follows, $ULC(x, z, p)\ [t_1 - t_2]\ E$, which summarizes the three base level commitments into one ULC such that now agent x commits p to agent z.

This aggregation type can be used in the following scenario: one worker commits to his foreman to produce a certain amount of goods by a particular time, the foreman in turn commits to his boss that a certain amount of goods will indeed be produced in the specified time. The ULC that summarizes these two commitments would be as follows: the worker commits to the boss to produce a certain amount of goods in the specified time.

Aggregation, by definition, causes low level (in our case base level) information loss. Although the upper level commitment has the equivalent effect/results of the base level commitments that constitute the upper level commitment at the cost of loosing some details in lower level commitments, the upper level commitment is in no way identical to the *summation* of the *individual* base level commitments. So, the result is only equivalent but not identical. While creating an upper level commitment, we may loose some of the details of the base level commitments that forms this upper level commitment. In case of transitive relations, for instance, by simply looking at the upper level commitment we may not be aware of the agents who were involved in the intermediate commitments. The organization should decide whether the lost information resulting from aggregation is important and whether to proceed with the aggregation.

The previous example is a logical one when we think the proposition p, as "paying USD 100". In the base level, x commits to pay to u, u in turn commits to pay to y, and finally y commits to pay to z. All of these transactions can be aggregated as simply x commits to pay to z, as there are inherent transitive relations among the base level commitments. But if we only look at the aggregated commitment, we may not know that apart from agents x and z, two other agents, u and y, are also involved in the process.

6.2. Combining Organizational Work-flows

As we may recall from earlier, a commitment protocol enlists the rules of interactions among agents in a multiagent organization. As such, an organization can be expressed by a commitment protocol. Two (or more) organizational work-flows may be combined, by combining their respective commitment protocols together, given that the agents – both *debtors* and *creditors*, *propositions*, and *time bounds* used in the specifications of these commitment protocols are logically related entities.

It is important to note that the uniqueness of entities is respected while combining. For example, if an agent, x, is used in commitments of the commitment protocols

to be combined, x should refer to the same agent over all commitment protocols. Likewise, propositions in commitments; and time points used to bind commitments are global. Moreover, all time points are linearly related.

Let A and B be two commitment protocols. Commitments of these two protocols are given in Table 6.1 along with their time bounds.

Table 6.1. Commitment protocols A and B.

Protocol A	**Protocol** B
$C_{A1}(Producer, GM, Produce)$ $[Q1]$ U	$C_{B1}(Producer, GM, Produce)$ $[Q1]$ U
$C_{A2}(Seller, GM, Sell)$ $[Q2]$ U	$C_{B2}(Developer, GM, Develop)$ $[Q1]$ U

In order to combine these two commitment protocols, we, first need to check if there is any *conflicting* commitments given time bounds. In this case, there is none. So, we can proceed with the task of combining these two commitment protocols. Next, we check for *duplicate* commitments among the commitment protocols. Duplicates are discarded if there is any. Here we can see that commitments C_{A1} and C_{B1} are duplicates. Only one will suffice in the resulting combined commitment protocol. We can discard either of them but not both and it is not really important which one we discard. The resulting combined commitment protocol is given in Table 6.2.

Table 6.2. Commitment protocols A and B combined together.

Combined Protocol
$C_1(Producer, GM, Produce)$ $[Q1]$ U
$C_2(Seller, GM, Sell)$ $[Q2]$ U
$C_3(Developer, GM, Develop)$ $[Q1]$ U

Now, let us add an additional commitment, $C_{B3}(Producer, GM, Not\ Produce)$ $[Q2]$ U, to commitment protocol B (See Table 6.3).

Although the proposition in commitments C_{A2} and C_{B3} are inverse (negation) of each other, these two commitments do not conflict given their time bounds. So, we can proceed with the task of combining as well as the previous case and get the following combined commitment protocol as given in Table 6.4.

Table 6.3. Commitment C_{B3} added to commitment protocol B.

Protocol A	**Protocol** B
$C_{A1}(Producer, GM, Produce)$ $[Q1]$ U	$C_{B1}(Producer, GM, Produce)$ $[Q1]$ U
$C_{A2}(Seller, GM, Sell)$ $[Q2]$ U	$C_{B2}(Developer, GM, Develop)$ $[Q1]$ U
	$C_{B3}(Producer, GM, Not\ Produce)$ $[Q2]$ U

Table 6.4. Commitment protocols A and B combined together.

Combined Protocol
$C_1(Producer, GM, Produce)$ $[Q1]$ U
$C_2(Seller, GM, Sell)$ $[Q2]$ U
$C_3(Developer, GM, Develop)$ $[Q1]$ U
$C_4(Producer, GM, Not\ Produce)$ $[Q2]$ U

Let us now add another commitment, $C_{A3}(Developer, GM, NotDevelop)$ $[Q1]$ U, to commitment protocol A as given in Table 6.5.

Table 6.5. Commitment C_{A3} added to commitment protocol A which is in conflict with commitment C_{B2}. Hence, these two commitment protocols cannot be combined.

Protocol A	**Protocol** B
$C_{A1}(Producer, GM, Produce)$ $[Q1]$ U	$C_{B1}(Producer, GM, Produce)$ $[Q1]$ U
$C_{A2}(Seller, GM, Sell)$ $[Q2]$ U	$C_{B2}(Developer, GM, Develop)$ $[Q1]$ U
$C_{A3}(Developer, GM, NotDevelop)$ $[Q1]$ U	$C_{B3}(Producer, GM, Not\ Produce)$ $[Q2]$ U

In this case, commitments C_{A3} and C_{B2} contains conflicting proposition and their time intervals are overlapping. Hence, these commitments are in conflict given their respective time bounds. Until and unless this conflict is resolved, we cannot combine these two commitments protocol.

This concludes all the concepts and techniques we have developed in order to represent the interactions in terms of commitments among agents in multiagent organization. These concepts are integrated in a software tool that helps commitment protocol designers to design and generate commitment protocols. In the following chapter we discuss the architecture of the tool in detail.

7. SOFTWARE ARCHITECTURE

The topics regarding commitments, commitment graph, time bounds of commitments, conflicts among commitments and their resolution, upper level aggregation of commitments, and combining commitment protocols that we have discussed so far in the previous chapters are implemented in a software tool. The tool is called "Protocol Validator". This tool helps us design consistent and conflict free commitment protocols, analyze the work-flow of a multiagent organization in terms of commitments, and create upper level abstraction of base level commitments in an interactive way. We have chosen to specify commitment protocols in XML format.

7.1. Protocol Specification in XML

The XML commitment specification file, first, contains the time points that are used in the protocol. The order of the time points are important as the linear ordering of time points is derived from the order they are in the protocol specification file.

Once the time points are specified, the protocol specification file contains the set of commitments. These commitments are generally time bound commitments as their validity is limited by specific time periods. A commitment element in the XML commitment protocol file contains the following information regarding a commitment: commitment's unique ID, debtor of the commitment, creditor of the commitment, proposition of the commitment, time interval of the commitment which is bounded by two time points, time quantifier of the time interval, and finally, the set of operations that this commitment can perform.

If we would like to put constraints between commitments or on propositions we have to specify those constraints in the XML commitment protocol file after enlisting all the commitments. Constraint between a pair of commitments specify if one commitment of the pair has to be valid before the other or they have to be valid at the same time. On the other hand, constraints on propositions specify what other

proposition has to hold for this proposition to hold at a particular time. Time points, commitments, and constraints appear in the commitment specification file as follows:

```
<protocol>
    <timePoints>
        <timePoint>tb</timePoint>
        <timePoint>t0</timePoint>
        <timePoint>te</timePoint>
    </timePoints>
    <commitments>
        <commitment>
          <commitmentId>C1</commitmentId>
          <debtor>x</debtor>
          <creditor>y</creditor>
          <proposition>p</proposition>
          <time>
              <from>t1</from>
              <to>t2</to>
              <quantifier>Existential</quantifier>
          </time>
          <operation>Cancel</operation>
          <operation>Release</operation>
      </commitment>
    </commitments>
    <constraints>
       <constraint left="C1" right="C2" relation="before"/>
    </constraints>
    <cnstsOnProp>
       <cnstOnProp prop="q" requires="p" relation="parallel"/>
    </cnstsOnProp>
</protocol>
```

7.2. Design of the Tool

The software tool "Protocol Validator", developed in Java, first reads protocol specification from an XML file which contains the time points in linear order and the commitments along with their time bounds. The protocol file may also contain constraints, if any, between commitments or on propositions. The constraints can assist us in resolving some of the conflicts or in labeling a conflict as unrealistic.

After reading the specification file, Protocol Validator generates a commitment graph from the commitments. The commitment graph is necessary in order to detect the commitments that cannot be resolved. Protocol Validator marks an unresolvable node as *black* which can be made *white* with the help of Protocol Validator by creating a link with this node to node RC.

Once the commitment graph is created, Protocol Validator proceeds to detect conflicting commitments if any. Conflicts are displayed and options are provided on how a particular conflict can be resolved. The user can use a suitable option to resolve a conflict. If a commitment is modified, the protocol designer can *re*-create the commitment graph to check whether a modification in a commitment creates a new conflict. This process advances iteratively until the protocol designer gets a consistent and conflict-free commitment protocol that adequately reflects the work-flow of an organization.

Protocol Validator also creates upper level commitments from the existing base level commitments and displays them. If a protocol is modified by Protocol Validator, the modified protocol can be saved in XML format for later use. Using Protocol Validator, two or more commitments can be combined together and the unified protocol can be saved as well.

7.3. Packages Designed in Java

The packages, designed to develop Protocol Validator are described below:

edu.boun.cmpe.cmt.time This package contains *TimePoint* class and *TimeQuantifier* interface. *TimePoint* class defines a time point. *TimeQuantifier* interface defines two time quantifiers: one is *universal* and the other is *existential*. *TimePointList* class is a collection that contains all the time points in linearly increasing order. *TimeInterval* class defines a time interval between two *TimePoint*s and is associated with a *TimeQuantifier*.

edu.boun.cmpe.cmt.operation This package contains *Operation* class and *OperationType* interface. *OperationType* interface defines the operations that can be used on commitments. *Operation* class defines an operation.

edu.boun.cmpe.cmt.commitment The core class in this package is the *Commitment* class from which several other classes are derived. *Commitment* class defines a base level commitment. *TimeBoundCommitment* class is derived from this class which defines a commitment with an associated *TimeInterval*. *ConflictType* interface defines the type of conflict between commitments given time bound. *ConflictingCommitments* class contains a pair of conflicting commitments. *ConflictingCommitmentsGivenTimeBound* is derived from this class which contains a pair of conflicting commitments, the time interval in which they conflict and the conflict type. Finally, *DuplicateCommitments* class contains a pair of duplicate commitments.

edu.boun.cmpe.cmt.commitment.ulc This package contains *UpperLevelCommitment* class and *CommitmentAggregation* interface. *UpperLevelCommitment* class contains an upper level commitment generated from a set of base level commitments. *CommitmentAggregation* interface defines the types of aggregation.

edu.boun.cmpe.cmt.graph This package contains *Node* class that defines a node created from a commitment, *DirectedEdge* class that defines a directed edge between nodes, *NodeColor* interface that defines node colors, and *Graph* class that contains functions to create a commitment graph. *Graph* class also contains the functions that detects conflicts and duplicates among commitments.

edu.boun.cmpe.cmt.constraint There are two classes in this package: *Constraint* class defines a constraint between a pair of commitments while *ConstraintOnProposition* class defines a constraint on a proposition.

In this chapter, we have discussed the software tool that we have developed in order to implement the concepts regarding commitment and time we have explained earlier. In doing so, we have shown that these concepts can be implemented and in the next chapter we describe a case study involving protocol specifications of two organizations that are combined in Protocol Validator.

8. A CASE STUDY

In this chapter, a detailed analysis of how an organization can be stated in terms of commitments is given. We envision a hypothetical business organization that produces and sells toys. Let us call this organization $OrgL$. Later we will analyze how this organization can be combined with a similar but smaller organization, called $OrgS$. $OrgS$ also produces toys but they do not sell directly to the market. $OrgS$ merely acts like a subcontractor and works for other larger organizations.

At the top of the organizational hierarchy of organization $OrgL$ is the General Manager (GM_{OrgL}) who delegates his duties through various department heads (DHs) of the organization. For simplicity, we will include a subset of the departments, which are Production (Pr), Internal Sales (Is), External Sales (Es) and Inventory (In). The Pr department of $OrgL$ and the DH of this department will denoted as Pr_{OrgL} and DH^{Pr}_{OrgL}, respectively. An employee (E) working in the Production department will be denoted as EPr^{id}_{OrgL}, where id is used to distinguish several employees. The commitments that we are going to specify here will be for a specific year which is divided into Quarters, denoted as (Q_i) where i is a quarter number.

Let us assume that there are two employees in Pr_{OrgL} who commit to their DH^{Pr}_{OrgL} to produce toys ($Produce$) for the first three quarters. The commitments of these employees are formally specified below.

C^1_{OrgL} (EPr^1_{OrgL}, DH^{Pr}_{OrgL}, $Produce$) [Q_1] U – Employee One of the Production department commits to the department head to produce toys in the first quarter.

Similarly for the other two quarters, the commitments are:

C^2_{OrgL} (EPr^1_{OrgL}, DH^{Pr}_{OrgL}, $Produce$) [Q_2] U

C^3_{OrgL} (EPr^1_{OrgL}, DH^{Pr}_{OrgL}, $Produce$) [Q_3] U

For the second employee in the production department (EPr^2_{OrgL}), the corresponding three commitments for four quarters are:

C^4_{OrgL} (EPr^2_{OrgL}, DH^{Pr}_{OrgL}, $Produce$) [Q_1] U

C^5_{OrgL} (EPr^2_{OrgL}, DH^{Pr}_{OrgL}, $Produce$) [Q_2] U

C^6_{OrgL} (EPr^2_{OrgL}, DH^{Pr}_{OrgL}, $Produce$) [Q_3] U

Please note that neither of these two employees makes any commitment to produce toys for the fourth quarter.

Internal Sales department has one employee (EIs^1_{OrgL}) for the domestic market who commits to the respective department head (DH^{Is}_{OrgL}) to sell toys ($Sell$) in each quarter. The corresponding four commitments for four quarters are:

C^7_{OrgL} (EIs^1_{OrgL}, DH^{Is}_{OrgL}, $Sell$) [Q_1] U

C^8_{OrgL} (EIs^1_{OrgL}, DH^{Is}_{OrgL}, $Sell$) [Q_2] U

C^9_{OrgL} (EIs^1_{OrgL}, DH^{Is}_{OrgL}, $Sell$) [Q_3] U

C^{10}_{OrgL} (EIs^1_{OrgL}, DH^{Is}_{OrgL}, $Sell$) [Q_4] U

And, the only employee (EEs^1_{OrgL}) in the external sales department commits to the respective department head (DH^{Es}_{OrgL}) to sell toys ($Sell$) in the overseas market in each quarter. The corresponding four commitments for four quarters are:

C^{11}_{OrgL} (EEs^1_{OrgL}, DH^{Es}_{OrgL}, $Sell$) [Q_1] U

C^{12}_{OrgL} (EEs^1_{OrgL}, DH^{Es}_{OrgL}, $Sell$) [Q_2] U

C^{13}_{OrgL} (EEs^{1}_{OrgL}, DH^{Es}_{OrgL}, $Sell$) [Q_3] U

C^{14}_{OrgL} (EEs^{1}_{OrgL}, DH^{Es}_{OrgL}, $Sell$) [Q_4] U

The employee in Inventory (EIn^{1}_{OrgL}) commits to the Inventory Department Head (DH^{In}_{OrgL}) to take inventory ($Take\ Inventory$ or TI in short) once in every quarter who in turn commits the same thing to the General Manager (GM_{OrgL}). The corresponding commitments are:

C^{15}_{OrgL} (EIn^{1}_{OrgL}, DH^{In}_{OrgL}, TI) [Q_1] E

C^{16}_{OrgL} (EIn^{1}_{OrgL}, DH^{In}_{OrgL}, TI) [Q_2] E

C^{17}_{OrgL} (EIn^{1}_{OrgL}, DH^{In}_{OrgL}, TI) [Q_3] E

C^{18}_{OrgL} (EIn^{1}_{OrgL}, DH^{In}_{OrgL}, TI) [Q_4] E

C^{19}_{OrgL} (DH^{In}_{OrgL}, GM_{OrgL}, TI) [Q_1] E

C^{20}_{OrgL} (DH^{In}_{OrgL}, GM_{OrgL}, TI) [Q_2] E

C^{21}_{OrgL} (DH^{In}_{OrgL}, GM_{OrgL}, TI) [Q_3] E

C^{22}_{OrgL} (DH^{In}_{OrgL}, GM_{OrgL}, TI) [Q_4] E

And finally, a contradictory commitment from the employee in the external sales department (EEs^{1}_{OrgL}) where it makes a commitment not to sell ($Not\ Sell$) the toys for the whole year! The idea here is to show how a conflict can be managed in Protocol Validator. This commitment is specified below:

C^{C}_{OrgL} (EEs^{1}_{OrgL}, DH^{Es}_{OrgL}, $Not\ Sell$) [Y] U, where $Y = Q_1 + Q_2 + Q_3 + Q_4$

When we feed Protocol Validator with this commitment protocol, conflicts among commitments are detected in line with our expectations. The commitment C^{C}_{OrgL} is in conflict with the commitments C^{11}_{OrgL}, C^{12}_{OrgL}, C^{13}_{OrgL}, and C^{14}_{OrgL} because EEs^{1}_{OrgL} is now making a commitment of *not selling* which contradicts with all of its earlier commitments of *selling* in the same time period.

The best way to deal with this conflict is to discard commitment C^{C}_{OrgL} as not only does it cause a total of four conflicts but also a commitment of *not selling* is not simply required. Similarly, we can shift the time bound of commitment C^{C}_{OrgL} such that the new time bound does not overlap with the year in question.

After discarding the contradictory commitment and processing the protocol again, we see that although there is no conflict in the protocol, some of the commitments are not *realistic*. C^{10}_{OrgL} and C^{14}_{OrgL} cannot be fulfilled as it violates the constraint which says that in order to *Sell* toys, there must be some commitment to *Produce* them in the same time interval. Currently there is no commitment to *Produce* toys in the fourth quarter. Therefore, any commitment to *Sell* toys in the fourth quarter would be unrealistic.

Analyzing remaining commitments further, we obtain a set of *Upper Level Commitments* (ULC). The first three ULCs are constructed by aggregation on debtors. The first ULC states that employees EPr^{1}_{OrgL} and EPr^{2}_{OrgL} together will produce toys in the first quarter which is equivalent to stating that the production department, Pr (comprising of two employees) now commits to DH^{Pr}_{OrgL} to *Produce* toys for the first quarter. The ULC is,

$ULC^{1}_{OrgL}\,(EPr^{1}_{OrgL}{+}EPr^{2}_{OrgL}, DH^{Pr}_{OrgL}, Produce)\,[Q_1]\,U\,\{C^{1}_{OrgL}, C^{4}_{OrgL}\}$ – aggregation on debtor of base level commitments, C^{1}_{OrgL} and C^{4}_{OrgL}.

Similarly, the other two ULCs by aggregating debtors are:

$ULC^{2}_{OrgL}\,(EPr^{1}_{OrgL} + EPr^{2}_{OrgL}, DH^{Pr}_{OrgL}, Produce)\,[Q_2]\,U\,\{C^{2}_{OrgL}, C^{5}_{OrgL}\}$

ULC^{3}_{OrgL} $(EPr^{1}_{OrgL} + EPr^{2}_{OrgL}, DH^{Pr}_{OrgL}, Produce)$ $[Q_3]$ U $\{C^{3}_{OrgL}, C^{6}_{OrgL}\}$

Then two ULCs by aggregating time (adding first three quarters) for these two employees separately in the Production department are obtained. Each employee commits to *Produce* for a period of first three quarters.

ULC^{4}_{OrgL} $(EPr^{1}_{OrgL}, DH^{Pr}_{OrgL}, Produce)$ $[Q_1 + Q_2 + Q_3]$ U $\{C^{1}_{OrgL}, C^{2}_{OrgL}, C^{3}_{OrgL}\}$

ULC^{5}_{OrgL} $(EPr^{2}_{OrgL}, DH^{Pr}_{OrgL}, Produce)$ $[Q_1 + Q_2 + Q_3]$ U $\{C^{4}_{OrgL}, C^{5}_{OrgL}, C^{6}_{OrgL}\}$

The Internal Sales employee (EIs^{1}_{OrgL}) is now expected to sell toys for the whole year.

ULC^{6}_{OrgL} $(EIs^{1}_{OrgL}, DH^{Is}_{OrgL}, Sell)$ $[Q_1 + Q_2 + Q_3 + Q_4]$ U $\{C^{7}_{OrgL}, C^{8}_{OrgL}, C^{9}_{OrgL}, C^{10}_{OrgL}\}$

Similarly, for the External Sales employee (EEs^{1}_{OrgL}) we obtain another ULC.

ULC^{7}_{OrgL} $(EEs^{1}_{OrgL}, DH^{Es}_{OrgL}, Sell)$ $[Q_1 + Q_2 + Q_3 + Q_4]$ U $\{C^{11}_{OrgL}, C^{12}_{OrgL}, C^{13}_{OrgL}, C^{14}_{OrgL}\}$

The Inventory employee (EIn^{1}_{OrgL}) commits to take inventory (TI) once in each quarter. So does the Inventory Department Head (DH^{In}_{OrgL}).

ULC^{8}_{OrgL} $(EIn^{1}_{OrgL}, DH^{In}_{OrgL}, TI)$ $[Q_1 + Q_2 + Q_3 + Q_4]$ E $\{C^{15}_{OrgL}, C^{16}_{OrgL}, C^{17}_{OrgL}, C^{18}_{OrgL}\}$

ULC^{9}_{OrgL} $(DH^{In}_{OrgL}, GM_{OrgL}, TI)$ $[Q_1 + Q_2 + Q_3 + Q_4]$ E $\{C^{19}_{OrgL}, C^{20}_{OrgL}, C^{21}_{OrgL}, C^{22}_{OrgL}\}$

And finally, four ULCs are created on transitivity of commitments. As the employee in the Inventory department (EIn^{1}_{OrgL}) commits to $Take\ Inventory$ to the department head (DH^{In}_{OrgL}) who in turn commits the same thing to the General Manager (GM_{OrgL}) for each quarter. These ULCs are:

ULC^{10}_{OrgL} $(EIn^{1}_{OrgL}, GM_{OrgL}, TI)$ $[Q_1]$ E $\{C^{15}_{OrgL}, C^{19}_{OrgL}\}$

ULC^{11}_{OrgL} $(EIn^{1}_{OrgL}, GM_{OrgL}, TI)$ $[Q_2]$ E $\{C^{16}_{OrgL}, C^{20}_{OrgL}\}$

ULC^{12}_{OrgL} $(EIn^{1}_{OrgL}, GM_{OrgL}, TI)$ $[Q_3]$ E $\{C^{17}_{OrgL}, C^{21}_{OrgL}\}$

ULC^{13}_{OrgL} $(EIn^{1}_{OrgL}, GM_{OrgL}, TI)$ $[Q_4]$ E $\{C^{18}_{OrgL}, C^{22}_{OrgL}\}$

Each of these ULCs now states that the employee in the Inventory department (EIn^{1}_{OrgL}) commits to the General Manager (GM_{OrgL}) to $Take\ Inventory$ in each quarter.

Cooperating with Another Organization: Since $OrgL$ will not be producing any toys in the fourth quarter, $OrgL$ wants to cooperate with $OrgS$ so that its selling of toys is not disrupted for the fourth quarter. $OrgS$ only produces toys but do not have the capability to sell them directly in the market. This makes sense for $OrgS$ too as its products will be sold by $OrgL$. They plan to cooperate from the beginning of third quarter.

This smaller organization has a General Manager (GM_{OrgS}), a Production department head (DH^{Pr}_{OrgS}) and two employees (EPr^{1}_{OrgS}) and (EPr^{2}_{OrgS}).

After the organizations are combined, these employees will be under the supervision of $OrgL$'s Production department head (DH^{Pr}_{OrgL}). The employees commit to their new department head to produce toys for the last two quarters of the year. The corresponding commitments are given below.

C^{1}_{OrgS} $(EPr^{1}_{OrgS}, DH^{Pr}_{OrgL}, Produce)$ $[Q_3]$ U

C^{2}_{OrgS} $(EPr^{1}_{OrgS}, DH^{Pr}_{OrgL}, Produce)$ $[Q_4]$ U

C^{3}_{OrgS} $(EPr^{2}_{OrgS}, DH^{Pr}_{OrgL}, Produce)$ $[Q_3]$ U

C^{4}_{OrgS} $(EPr^{2}_{OrgS}, DH^{Pr}_{OrgL}, Produce)$ $[Q_4]$ U

Two sets of ULCs can be obtained from these four commitments. The first set of ULCs are based on aggregation of debtors where both employees as a group commits to *Produce* to DH^{Pr}_{OrgL} in each of the last two quarters.

ULC^1_{OrgS} $(EPr^1_{OrgS} + EPr^2_{OrgS}, DH^{Pr}_{OrgL}, Produce)$ $[Q_3]$ U $\{C^1_{OrgS}, C^3_{OrgS}\}$

ULC^2_{OrgS} $(EPr^1_{OrgS} + EPr^2_{OrgS}, DH^{Pr}_{OrgL}, Produce)$ $[Q_4]$ U $\{C^2_{OrgS}, C^4_{OrgS}\}$

The second set of ULCs are based on time aggregation where each employee commits to *Produce* for two quarters combined.

ULC^3_{OrgS} $(EPr^1_{OrgS}, DH^{Pr}_{OrgL}, Produce)$ $[Q_3 + Q_4]$ U $\{C^1_{OrgS}, C^2_{OrgS}\}$

ULC^4_{OrgS} $(EPr^2_{OrgS}, DH^{Pr}_{OrgL}, Produce)$ $[Q_3 + Q_4]$ U $\{C^3_{OrgS}, C^4_{OrgS}\}$

When we combine these two organizations, the first thing we expect to see and we do see that the unrealistic commitments in $OrgL$ becomes realistic as substitute productions capability is added for fourth quarter. Therefore, the employees in the sales departments can now continue selling toys for the fourth quarter.

As these two organizations are combined together, we obtain a slightly different set of ULCs acknowledging the fact that two sets of commitments are now used together to obtain these ULCs. This time the commitments of the employees of $OrgS$ are also taken into account.

The commitment protocols specifying the work-flows of organizations $OrgL$ and $OrgS$ in XML format are given in APPENDIX A and APPENDIX B, respectively. In APPENDIX C, included is the combined commitment protocol.

9. DISCUSSION

In this thesis, we have developed a commitment-based approach for representing inter-agent communications in a multiagent organization. In this approach, a set of commitments forms a commitment protocol which is used to represent the work-flow of a multiagent organization.

Our work builds on some previous work on this field, notably [5, 10], where the concept of commitments and operations on commitments are presented in detail. In [5], a set of algorithms is presented that helps us detect conflicts among commitments and check for consistency and correctness in the commitment protocol. These algorithms first create a commitment graph and then detects unresolvable commitments in the commitment protocol. However, the commitments used in these algorithms do not accommodate the notion of time.

For a commitment protocol to represent the work-flow of multiagent organizations in a realistic way, the concept of time needs to be associated with commitments. This implies that commitments should be bound by a finite period of time. In our work, we have developed the idea of time bound commitments where commitments remain valid for a specified finite period of time. In doing so, we have introduced a number of key concepts related to the notion of time. In our model, time is represented as a collection of linearly ordered discrete *time points*. Two time points form a *time interval*, and we use this time interval to indicate the period of time in which a commitment is expected to be valid. Time intervals are quantified by one of the two *time quantifiers* – *universal* quantifier requires that the proposition of a commitment holds along the entire time interval whereas *existential* quantifier requires the proposition to hold at least once during the entire time interval.

We also introduce the *transition point* that facilitates conflict-free reversal of a proposition at a particular time point. We redefine the term *conflict* among commitments as the association of time with commitments may render some previously

detected conflicts (without taking into account the notion of time) no longer in conflict.

Once commitments are bounded with time intervals, we modify the existing algorithms and develop a new algorithm to deal with the notion of time while detecting conflicts among commitments. When a pair of time bound commitments does not have an overlapping time interval, there is no scope for conflict. Therefore, the set of conflicts among time bound commitments is intuitively a subset of the set of conflicts among commitments when time periods are not considered.

We have discussed all of the main conflict scenarios that may arise among commitments in a commitment protocol representing the work-flow of a multiagent organization. These scenarios primarily differ on the basis of the overlapping section of two time intervals. These are *identical*, *containing*, and *intersecting* time intervals. In some instances a conflict between a pair of commitments is simply a *possibility* while in other instances a conflict is *certain* to occur. Time quantifiers of the commitments in a conflict are used to judge whether a conflict is a mere possibility or the conflict is a certainty.

In our work, we have formulated a number of strategies to deal with conflicts among time bound commitments and ways to resolve them, if possible. The simplest way is to get rid of a commitment that causes conflict. But this approach might prevent us from representing a multiagent organization in a realistic manner. Hence, instead of discarding a commitment altogether from a commitment protocol, we may use two other strategies for resolving conflicts – modifying the composition of a commitment or shifting time intervals of commitments. We can also use *constraints* between commitments such that two conflicting commitments do not have to be valid during the same time period by stating that for a commitment to be valid the other commitment must be fulfilled and hence cease to exist.

Although a commitment may not be in conflict, it, nonetheless, may be unrealistic if it depends on other commitment to be fulfilled before this commitment can be fulfilled

and the other commitment on which this commitment depends on is not fulfilled or does not exist at all.

We have also discussed how we can create an upper level abstraction of commitments by grouping related commitments so that we have a concise view of the work-flow of a multiagent organization. Upper level commitments reflect the inherent logical similarities among lower level commitments. Upper level abstraction of commitments can be created by aggregating time intervals of similar commitments, by grouping agents who make same propositions, by grouping commitments that are part of a larger conceptual commitment, and by capturing the inherent transitive relation of a number of commitments.

When we create an upper level abstraction of commitments we might loose some base level details as *aggregation*, by definition, may cause some low level information loss. It is up to the commitment protocol designer whether to proceed with the abstraction in the face of information loss.

We have shown a way to merge multiple work-flows of multiagent organizations by integrating their individual commitment protocols. This reflects real-life situations when two organizations merge with each other.

To supplement out theoretical work, we have developed a software tool in Java that can help a commitment protocol designer to design a sound and consistent commitment protocol. Commitment protocols, specified in XML format, are fed to this tool and the tool detects inconsistencies and conflicts among commitments, if any. The commitment protocol developer can deal with these inconsistencies and conflicts in an interactive way with the tool and this process iterates until the protocol designer gets a set of consistent and conflict-free commitments that satisfactorily meets the design requirements. The tool also creates and displays the upper level commitments from the existing base level commitments.

9.1. Literature Survey

In [10], Singh presents a rich definition of social commitments that motivates an architecture for multiagent systems. He calls this *spheres of commitment*. He identifies key operations on commitments and multiagent systems. Multiagent systems, viewed as spheres of commitment, provide the context for the different operations on commitments. He synthesizes ideas from multiagent systems, especially that of social context, with the ideas of ethics and legal reasoning by capturing normative concepts such as obligations, taboos, conventions, and pledges as different kind of commitments.

In [5], Yolum develops and formalizes design requirements for developing correct and consistent commitment protocols. She establishes and applies correctness properties on commitment protocols, and designs a number of algorithms that checks for correctness and consistency in commitment protocols. A correct commitment protocol defines the necessary interactions among agents to lead a multiagent system to its desired state. Our thesis primarily builds on the concepts and algorithms presented here.

Although the algorithms described here for checking consistency and correctness of commitment protocols, do not take the notion of time into account, we in our work incorporate the concept of time in order to realistically specify commitments.

Fornara and Colombetti [7] develop a method for agent communication based on the social notion of commitments where commitments are defined operationally within an object-oriented paradigm. They formalize operational specification of commitment class as an abstract data type and they use commitment objects (an instance of commitment class) to define the meaning of speech acts. In this way they can verify the soundness of conversational protocols.

Their approach in defining a standard agent communication language (ACL) is based on speech act theory [14] which views language use as a form of action, making it possible to treat communicative acts and other types of actions in a uniform way.

Speech act theory appears so suitable to describe communicative interactions among artificial agents that almost all existing proposals for ACLs are based on it. Furthermore, as it provides an adequate approach to human communication, speech act theory allows the successful mixed interactions among human beings and software agents.

In their work, the characteristics, structures and the evolution of commitments through time are analyzed. They discuss further on the concepts of social commitment, conditional commitment, precommitment, and temporal proposition. Although they define commitment-based semantics for inter-agent communication, they do not elaborate on any possible inconsistency that may arise in the protocol and how to resolve this inconsistency. Nor do they use any time frame for commitments that would indicate how long a commitment should continue to persist.

Artikis *et al.* [8] develop a framework to specify, animate, and reason about open computational societies. They provide a formal framework for executable specifications of open computational societies. A computational society is *open* if it has following properties: 1) the behavior of the members and their interactions cannot be predicted in advance, 2) the internal architecture of the agents is not publicly known, and 3) the member of the society do not necessarily have common goals, desires or intentions.

They also identify a number of concepts that characterize an open agent society. A society is characterized by a set of agents, a set of constraints on the society (such as norms), a set of roles that members can play, the state of the members and the environment in which they act, a communication language, relationship between the members, and the structure of the society. Among these concepts, they focus more on social constraints, social roles, and social states. They make use of event calculus in their approach of identifying the concepts of computational societies, such as social roles and social states. But they do not provide any design rules to establish the correctness of the executed societies.

Chopra and Singh [15] formulate a method to generate new protocols from existing protocols by applying transformers on existing protocols. The main idea here is

that as an existing protocol may not exactly fit all the possible different circumstances that may arise in different contexts, the existing protocol needs to be transformed to handle different contexts (i.e., circumstances). In this sense a protocol is said to be contextualized to ensure correct interactions that might depend upon context.

Developing protocols from scratch for each of the possible contexts is an enormous, difficult, tiresome and error prone task. Instead, transformers can be used on existing protocols when and where applicable in order to considerably reduce and simplify the work load on part of the protocol designer. They specify protocols and transformers formally and declaratively where each transformer may be designed for a particular aspect of context.

They use a purchase protocol as an example and show how different variations of the purchase act can be handled when the protocol is transformed in different ways by composing its specification with different transformer specifications. They specify one transformer that accommodates reminders; another that accommodates returns and refunds. One of the problems that may arise is that transforming protocols in this way may result in conflict between the original protocol and its transformer. In addition there is no guarantee that the resulting protocol will always allow meaningful transactions.

Mallya, Yolum, and Singh [12] develop a representation for temporal content of commitments in order to capture implicit temporal references and avoid ambiguities. They propose a temporal framework along with temporal qualification on commitments. They also provide the grammar – expressed in Backus-Naur Form, and semantics for the formal language to describe their scheme. Furthermore, they focus on commitment life cycle (from creation to breach or from creation to satisfaction) and resolving temporal commitments. A commitment is resolvable if its satisfaction or breach can be determined at some point. They develop methods to detect resolvability of temporally quantified propositions. They introduce time intervals and temporal anaphora to deal with time periods which is usual in real-life business deals. However, they do not consider finding conflicts between commitments as we have done here.

In [6], Singh discusses how commitments relate to various aspects of agents and what roles they play in multiagent systems. His approach relies on social concepts to formalize commitments. He presents the unifying principles behind commitment for single-agent and multiagent systems that are based on database-specific ramifications through a synthesis of databases and distributed computing. This synthesis involves a generalization of *sphere of control* which seeks to characterize activities more generally than database transactions as the limitations of traditional transactions are not suitable for open environments.

In his work, Singh primarily focuses on 1) commitment and spheres of control in databases, 2) commitment in multiagent systems and its connection with group structures, and 3) formalization of commitments that explicitly considers context. He discusses social actions in order to elaborate operations on commitments and logical form of commitments.

Dignum, Vázquez-Salceda, and Dignum [16] propose a framework, called OMNI (Organizational Model for Normative Institutions), for modeling different multiagent organizations – ranging from closed systems to open, flexible environments. OMNI allows a balance between the global organizational requirements and the autonomy of individual agents by integrating norms and contextual meanings of interactions among agents in one framework. Both the norms that regulate interaction between agents and the contextual meaning of these interactions are important aspects for specifying organizational structure.

OMNI is composed of three dimensions: Normative, Organizational and Ontological that characterize the environment and the framework is organized into three levels of abstraction: Abstract, Concrete and Implementation Level. It provides a formal logical semantics for all of its dimensions in order to ensure consistency. The modular structure of OMNI facilitates the adaptation of the framework to different types of domains. In those domains, with none or small normative components, design is guided by the Organizational dimension, while in highly regulated domains the Normative dimension is more important and thus affects the design. Our work differs

from OMNI as we use commitmnet-based approach where as ONMI is based on social and normative concepts such as organization structures, norms and domain language.

Alberti *et al.* [17] develop a formalism to specify constraints on agent interaction and a tool that is able to observe and check for agent compliance to interaction protocols. The tool is developed in Java and Prolog that utilizes logic programming technology. Their work is devoted to testing the compliance of agents to social rules, without having any knowledge on the internals of the agents. They provide a language, based on logics, to define the interaction protocols, and a proof-procedure, based on abduction, to check the compliance. Our work is different from this one in the sense that our main concern is to generate consistent and correct commitment protocol for a multiagent organization whereas this work is mainly concerned with the protocol run and compliance issue.

9.2. Future Directions

In our work, we have focused on introducing time and generating correct commitment protocol specification during compile time. A logical extension of our work would be to verify commitment protocols during run time. The protocol representation should allow the detection of non-compliant agents. By keeping track of the commitments that are created and resolved in the system, any violations of the protocol together with the violating agent can be detected.

Organization state transition can be incorporated into the system along with the time line such that when a commitment is discharged or canceled, among others, the organization state should change to another state from current state. By observing state transitions, a commitment's life cycle can be tracked to see if it is being fulfilled or violated. At the same time as agents fulfill or violate their commitments, complaint and non-compliant agents can be detected. At the end of the protocol run, the designer then can check whether the commitment protocol works as intended.

Currently, we are not equipped to process quantifiable information in commitments in general, and in propositions in particular. Two propositions of *producing* 100 *items* should amount to a single proposition of *producing* 200 *items*. On the contrary, two propositions of *turning on a particular light* should amount to a single proposition of *turning on a particular light once*.

The software tool provides suggestions, in a very naïve and wholesale way, on how a conflict can be resolved. Contemporary AI techniques can be used to prioritize conflict resolution strategies, for instance, finding a non-conflicting time interval for a conflicting commitment.

These are some interesting directions that we differ to future work.

APPENDIX A: COMMITMENT PROTOCOL OF $OrgL$

```
<?xml version='1.0' encoding='utf-8'?>
<protocol>
 <timePoints>
  <timePoint>tb</timePoint>
  <timePoint>Q1_B</timePoint>
  <timePoint>Q1_E</timePoint>
  <timePoint>Q1_2</timePoint>
  <timePoint>Q2_B</timePoint>
  <timePoint>Q2_E</timePoint>
  <timePoint>Q2_3</timePoint>
  <timePoint>Q3_B</timePoint>
  <timePoint>Q3_E</timePoint>
  <timePoint>Q3_4</timePoint>
  <timePoint>Q4_B</timePoint>
  <timePoint>Q4_E</timePoint>
  <timePoint>te</timePoint>
 </timePoints>

 <!--
  OrgL: Large Organization
  GML: General Manager of OrgL
  Departments          Department Head  Employees
  ...........          ..............  .........
  Pr: Production       DPrL            EPrL1, EPrL2
  Is: Internal Sales   DIsL            EIsL1, EIsL2
  Es: External Sales   DEsL            EEsL1, EEsL2
  In: Inventory        DInL            EInL1, EInL2
  Propositions
   Produce: Produce Toys of Type A
```

```
  Sell:    Sell Toys of Type A
  TI:      Take Inventory
-->

<commitments>
 <commitment>
  <commitmentId>CL1</commitmentId>
  <debtor>EPrL1</debtor>
  <creditor>DPrL</creditor>
  <proposition>Produce</proposition>
  <time>
   <from>Q1_B</from>
   <to>Q1_2</to>
   <quantifier>Universal</quantifier>
  </time>
  <operation>Discharge</operation>
 </commitment>
 <commitment>
  <commitmentId>CL2</commitmentId>
  <debtor>EPrL1</debtor>
  <creditor>DPrL</creditor>
  <proposition>Produce</proposition>
  <time>
   <from>Q1_2</from>
   <to>Q2_3</to>
   <quantifier>Universal</quantifier>
  </time>
 <operation>Discharge</operation>
 </commitment>
 <commitment>
  <commitmentId>CL3</commitmentId>
  <debtor>EPrL1</debtor>
```

```
 <creditor>DPrL</creditor>
 <proposition>Produce</proposition>
 <time>
  <from>Q2_3</from>
  <to>Q3_4</to>
  <quantifier>Universal</quantifier>
 </time>
 <operation>Discharge</operation>
</commitment>
<commitment>
 <commitmentId>CL4</commitmentId>
 <debtor>EPrL2</debtor>
 <creditor>DPrL</creditor>
 <proposition>Produce</proposition>
 <time>
  <from>Q1_B</from>
  <to>Q1_2</to>
  <quantifier>Universal</quantifier>
 </time>
 <operation>Discharge</operation>
</commitment>
<commitment>
 <commitmentId>CL5</commitmentId>
 <debtor>EPrL2</debtor>
 <creditor>DPrL</creditor>
 <proposition>Produce</proposition>
 <time>
  <from>Q1_2</from>
  <to>Q2_3</to>
  <quantifier>Universal</quantifier>
 </time>
 <operation>Discharge</operation>
```

```
</commitment>
<commitment>
 <commitmentId>CL6</commitmentId>
 <debtor>EPrL2</debtor>
 <creditor>DPrL</creditor>
 <proposition>Produce</proposition>
 <time>
  <from>Q2_3</from>
  <to>Q3_4</to>
  <quantifier>Universal</quantifier>
 </time>
 <operation>Discharge</operation>
</commitment>
<commitment>
 <commitmentId>CL7</commitmentId>
 <debtor>EIsL1</debtor>
 <creditor>DIsL</creditor>
 <proposition>Sell</proposition>
 <time>
  <from>Q1_B</from>
  <to>Q1_2</to>
  <quantifier>Universal</quantifier>
 </time>
 <operation>Discharge</operation>
</commitment>
<commitment>
 <commitmentId>CL8</commitmentId>
 <debtor>EIsL1</debtor>
 <creditor>DIsL</creditor>
 <proposition>Sell</proposition>
 <time>
  <from>Q1_2</from>
```

```
  <to>Q2_3</to>
  <quantifier>Universal</quantifier>
 </time>
 <operation>Discharge</operation>
</commitment>
<commitment>
 <commitmentId>CL9</commitmentId>
 <debtor>EIsL1</debtor>
 <creditor>DIsL</creditor>
 <proposition>Sell</proposition>
 <time>
  <from>Q2_3</from>
  <to>Q3_4</to>
  <quantifier>Universal</quantifier>
 </time>
 <operation>Discharge</operation>
</commitment>
<commitment>
 <commitmentId>CL10</commitmentId>
 <debtor>EIsL1</debtor>
 <creditor>DIsL</creditor>
 <proposition>Sell</proposition>
 <time>
  <from>Q3_4</from>
  <to>Q4_E</to>
  <quantifier>Universal</quantifier>
 </time>
 <operation>Discharge</operation>
</commitment>
<commitment>
 <commitmentId>CL11</commitmentId>
 <debtor>EEsL1</debtor>
```

```
 <creditor>DEsL</creditor>
 <proposition>Sell</proposition>
 <time>
  <from>Q1_B</from>
  <to>Q1_2</to>
  <quantifier>Universal</quantifier>
 </time>
 <operation>Discharge</operation>
</commitment>
<commitment>
 <commitmentId>CL12</commitmentId>
 <debtor>EEsL1</debtor>
 <creditor>DEsL</creditor>
 <proposition>Sell</proposition>
 <time>
  <from>Q1_2</from>
  <to>Q2_3</to>
  <quantifier>Universal</quantifier>
 </time>
 <operation>Discharge</operation>
</commitment>
<commitment>
 <commitmentId>CL13</commitmentId>
 <debtor>EEsL1</debtor>
 <creditor>DEsL</creditor>
 <proposition>Sell</proposition>
 <time>
  <from>Q2_3</from>
  <to>Q3_4</to>
  <quantifier>Universal</quantifier>
 </time>
 <operation>Discharge</operation>
```

```
</commitment>
<commitment>
 <commitmentId>CL14</commitmentId>
 <debtor>EEsL1</debtor>
 <creditor>DEsL</creditor>
 <proposition>Sell</proposition>
 <time>
  <from>Q3_4</from>
  <to>Q4_E</to>
  <quantifier>Universal</quantifier>
 </time>
 <operation>Discharge</operation>
</commitment>
<commitment>
 <commitmentId>CLC</commitmentId>
 <debtor>EEsL1</debtor>
 <creditor>DEsL</creditor>
 <proposition>-Sell</proposition>
 <time>
  <from>Q1_B</from>
  <to>Q4_E</to>
  <quantifier>Universal</quantifier>
 </time>
 <operation>Discharge</operation>
</commitment>
<commitment>
 <commitmentId>CL15</commitmentId>
 <debtor>EInL1</debtor>
 <creditor>DInL</creditor>
 <proposition>TI</proposition>
 <time>
  <from>Q1_B</from>
```

```
  <to>Q1_2</to>
  <quantifier>Existential</quantifier>
 </time>
 <operation>Discharge</operation>
</commitment>
<commitment>
 <commitmentId>CL16</commitmentId>
 <debtor>EInL1</debtor>
 <creditor>DInL</creditor>
 <proposition>TI</proposition>
 <time>
  <from>Q1_2</from>
  <to>Q2_3</to>
  <quantifier>Existential</quantifier>
 </time>
 <operation>Discharge</operation>
</commitment>
<commitment>
 <commitmentId>CL17</commitmentId>
 <debtor>EInL1</debtor>
 <creditor>DInL</creditor>
 <proposition>TI</proposition>
 <time>
  <from>Q2_3</from>
  <to>Q3_4</to>
  <quantifier>Existential</quantifier>
 </time>
 <operation>Discharge</operation>
</commitment>
<commitment>
 <commitmentId>CL18</commitmentId>
 <debtor>EInL1</debtor>
```

```
 <creditor>DInL</creditor>
 <proposition>TI</proposition>
 <time>
  <from>Q3_4</from>
  <to>Q4_E</to>
  <quantifier>Existential</quantifier>
 </time>
 <operation>Discharge</operation>
</commitment>
<commitment>
 <commitmentId>CL19</commitmentId>
 <debtor>DInL</debtor>
 <creditor>GML</creditor>
 <proposition>TI</proposition>
 <time>
  <from>Q1_B</from>
  <to>Q1_2</to>
  <quantifier>Existential</quantifier>
 </time>
 <operation>Discharge</operation>
</commitment>
<commitment>
 <commitmentId>CL20</commitmentId>
 <debtor>DInL</debtor>
 <creditor>GML</creditor>
 <proposition>TI</proposition>
 <time>
  <from>Q1_2</from>
  <to>Q2_3</to>
  <quantifier>Existential</quantifier>
 </time>
 <operation>Discharge</operation>
```

```
  </commitment>
  <commitment>
   <commitmentId>CL21</commitmentId>
   <debtor>DInL</debtor>
   <creditor>GML</creditor>
   <proposition>TI</proposition>
   <time>
    <from>Q2_3</from>
    <to>Q3_4</to>
    <quantifier>Existential</quantifier>
   </time>
   <operation>Discharge</operation>
  </commitment>
  <commitment>
   <commitmentId>CL22</commitmentId>
   <debtor>DInL</debtor>
   <creditor>GML</creditor>
   <proposition>TI</proposition>
   <time>
    <from>Q3_4</from>
    <to>Q4_E</to>
    <quantifier>Existential</quantifier>
   </time>
   <operation>Discharge</operation>
  </commitment>
 </commitments>

 <constraintsOnProposition>
  <constOnProp prop="Sell" req="Produce" relation="para"></constOnProp>
 </constraintsOnProposition>

</protocol>
```

APPENDIX B: COMMITMENT PROTOCOL OF $OrgS$

```
<?xml version='1.0' encoding='utf-8'?>
<protocol>
 <timePoints>
  <timePoint>tb</timePoint>
  <timePoint>Q1_B</timePoint>
  <timePoint>Q1_E</timePoint>
  <timePoint>Q1_2</timePoint>
  <timePoint>Q2_B</timePoint>
  <timePoint>Q2_E</timePoint>
  <timePoint>Q2_3</timePoint>
  <timePoint>Q3_B</timePoint>
  <timePoint>Q3_E</timePoint>
  <timePoint>Q3_4</timePoint>
  <timePoint>Q4_B</timePoint>
  <timePoint>Q4_E</timePoint>
  <timePoint>te</timePoint>
 </timePoints>

 <!--
  OrgS: Small Organization
  GML: General Manager of OrgS
  Departments          Department Head  Employees
  ...........          ..............  .........
  Pr: Production       DPrS            EPrS1, EPrS2
  Propositions
   Produce: Produce Toys of Type A
 -->

 <commitments>
```

```
<commitment>
 <commitmentId>CS1</commitmentId>
 <debtor>EPrS1</debtor>
 <creditor>DPrL</creditor>
 <proposition>Produce</proposition>
 <time>
  <from>Q2_3</from>
  <to>Q3_4</to>
  <quantifier>Universal</quantifier>
 </time>
 <operation>Discharge</operation>
</commitment>
<commitment>
 <commitmentId>CS2</commitmentId>
 <debtor>EPrS1</debtor>
 <creditor>DPrL</creditor>
 <proposition>Produce</proposition>
 <time>
  <from>Q3_4</from>
  <to>Q4_E</to>
  <quantifier>Universal</quantifier>
 </time>
 <operation>Discharge</operation>
</commitment>
<commitment>
 <commitmentId>CS3</commitmentId>
 <debtor>EPrS2</debtor>
 <creditor>DPrL</creditor>
 <proposition>Produce</proposition>
 <time>
  <from>Q2_3</from>
  <to>Q3_4</to>
```

```
    <quantifier>Universal</quantifier>
   </time>
   <operation>Discharge</operation>
  </commitment>
  <commitment>
   <commitmentId>CS4</commitmentId>
   <debtor>EPrS2</debtor>
   <creditor>DPrL</creditor>
   <proposition>Produce</proposition>
   <time>
    <from>Q3_4</from>
    <to>Q4_E</to>
    <quantifier>Universal</quantifier>
   </time>
   <operation>Discharge</operation>
  </commitment>
 </commitments>

</protocol>
```

APPENDIX C: AFTER COMBINING $OrgL$ AND $OrgS$

```
<?xml version='1.0' encoding='utf-8'?>
<!-- Created by Commitment Protocol Validator -->
<protocol>
 <timePoints>
  <timePoint>tb</timePoint>
  <timePoint>Q1_B</timePoint>
  <timePoint>Q1_E</timePoint>
  <timePoint>Q1_2</timePoint>
  <timePoint>Q2_B</timePoint>
  <timePoint>Q2_E</timePoint>
  <timePoint>Q2_3</timePoint>
  <timePoint>Q3_B</timePoint>
  <timePoint>Q3_E</timePoint>
  <timePoint>Q3_4</timePoint>
  <timePoint>Q4_B</timePoint>
  <timePoint>Q4_E</timePoint>
  <timePoint>te</timePoint>
 </timePoints>

 <commitments>
  <commitment>
   <commitmentId>CL1</commitmentId>
   <debtor>EPrL1</debtor>
   <creditor>DPrL</creditor>
   <proposition>Produce</proposition>
   <time>
    <from>Q1_B</from>
    <to>Q1_2</to>
    <quantifier>Universal</quantifier>
```

```
 </time>
 <operation>Discharge</operation>
</commitment>
<commitment>
 <commitmentId>CL2</commitmentId>
 <debtor>EPrL1</debtor>
 <creditor>DPrL</creditor>
 <proposition>Produce</proposition>
 <time>
  <from>Q1_2</from>
  <to>Q2_3</to>
  <quantifier>Universal</quantifier>
 </time>
<operation>Discharge</operation>
</commitment>
<commitment>
 <commitmentId>CL3</commitmentId>
 <debtor>EPrL1</debtor>
 <creditor>DPrL</creditor>
 <proposition>Produce</proposition>
 <time>
  <from>Q2_3</from>
  <to>Q3_4</to>
  <quantifier>Universal</quantifier>
 </time>
 <operation>Discharge</operation>
</commitment>
<commitment>
 <commitmentId>CL4</commitmentId>
 <debtor>EPrL2</debtor>
 <creditor>DPrL</creditor>
 <proposition>Produce</proposition>
```

```
 <time>
  <from>Q1_B</from>
  <to>Q1_2</to>
  <quantifier>Universal</quantifier>
 </time>
 <operation>Discharge</operation>
</commitment>
<commitment>
 <commitmentId>CL5</commitmentId>
 <debtor>EPrL2</debtor>
 <creditor>DPrL</creditor>
 <proposition>Produce</proposition>
 <time>
  <from>Q1_2</from>
  <to>Q2_3</to>
  <quantifier>Universal</quantifier>
 </time>
 <operation>Discharge</operation>
</commitment>
<commitment>
 <commitmentId>CL6</commitmentId>
 <debtor>EPrL2</debtor>
 <creditor>DPrL</creditor>
 <proposition>Produce</proposition>
 <time>
  <from>Q2_3</from>
  <to>Q3_4</to>
  <quantifier>Universal</quantifier>
 </time>
 <operation>Discharge</operation>
</commitment>
<commitment>
```

```
 <commitmentId>CL7</commitmentId>
 <debtor>EIsL1</debtor>
 <creditor>DIsL</creditor>
 <proposition>Sell</proposition>
 <time>
  <from>Q1_B</from>
  <to>Q1_2</to>
  <quantifier>Universal</quantifier>
 </time>
 <operation>Discharge</operation>
</commitment>
<commitment>
 <commitmentId>CL8</commitmentId>
 <debtor>EIsL1</debtor>
 <creditor>DIsL</creditor>
 <proposition>Sell</proposition>
 <time>
  <from>Q1_2</from>
  <to>Q2_3</to>
  <quantifier>Universal</quantifier>
 </time>
 <operation>Discharge</operation>
</commitment>
<commitment>
 <commitmentId>CL9</commitmentId>
 <debtor>EIsL1</debtor>
 <creditor>DIsL</creditor>
 <proposition>Sell</proposition>
 <time>
  <from>Q2_3</from>
  <to>Q3_4</to>
  <quantifier>Universal</quantifier>
```

```
 </time>
 <operation>Discharge</operation>
</commitment>
<commitment>
 <commitmentId>CL10</commitmentId>
 <debtor>EIsL1</debtor>
 <creditor>DIsL</creditor>
 <proposition>Sell</proposition>
 <time>
  <from>Q3_4</from>
  <to>Q4_E</to>
  <quantifier>Universal</quantifier>
 </time>
 <operation>Discharge</operation>
</commitment>
<commitment>
 <commitmentId>CL11</commitmentId>
 <debtor>EEsL1</debtor>
 <creditor>DEsL</creditor>
 <proposition>Sell</proposition>
 <time>
  <from>Q1_B</from>
  <to>Q1_2</to>
  <quantifier>Universal</quantifier>
 </time>
 <operation>Discharge</operation>
</commitment>
<commitment>
 <commitmentId>CL12</commitmentId>
 <debtor>EEsL1</debtor>
 <creditor>DEsL</creditor>
 <proposition>Sell</proposition>
```

```
 <time>
  <from>Q1_2</from>
  <to>Q2_3</to>
  <quantifier>Universal</quantifier>
 </time>
 <operation>Discharge</operation>
</commitment>
<commitment>
 <commitmentId>CL13</commitmentId>
 <debtor>EEsL1</debtor>
 <creditor>DEsL</creditor>
 <proposition>Sell</proposition>
 <time>
  <from>Q2_3</from>
  <to>Q3_4</to>
  <quantifier>Universal</quantifier>
 </time>
 <operation>Discharge</operation>
</commitment>
<commitment>
 <commitmentId>CL14</commitmentId>
 <debtor>EEsL1</debtor>
 <creditor>DEsL</creditor>
 <proposition>Sell</proposition>
 <time>
  <from>Q3_4</from>
  <to>Q4_E</to>
  <quantifier>Universal</quantifier>
 </time>
 <operation>Discharge</operation>
</commitment>
<commitment>
```

```
 <commitmentId>CL15</commitmentId>
 <debtor>EInL1</debtor>
 <creditor>DInL</creditor>
 <proposition>TI</proposition>
 <time>
  <from>Q1_B</from>
  <to>Q1_2</to>
  <quantifier>Existential</quantifier>
 </time>
 <operation>Discharge</operation>
</commitment>
<commitment>
 <commitmentId>CL16</commitmentId>
 <debtor>EInL1</debtor>
 <creditor>DInL</creditor>
 <proposition>TI</proposition>
 <time>
  <from>Q1_2</from>
  <to>Q2_3</to>
  <quantifier>Existential</quantifier>
 </time>
 <operation>Discharge</operation>
</commitment>
<commitment>
 <commitmentId>CL17</commitmentId>
 <debtor>EInL1</debtor>
 <creditor>DInL</creditor>
 <proposition>TI</proposition>
 <time>
  <from>Q2_3</from>
  <to>Q3_4</to>
  <quantifier>Existential</quantifier>
```

```
 </time>
 <operation>Discharge</operation>
</commitment>
<commitment>
 <commitmentId>CL18</commitmentId>
 <debtor>EInL1</debtor>
 <creditor>DInL</creditor>
 <proposition>TI</proposition>
 <time>
  <from>Q3_4</from>
  <to>Q4_E</to>
  <quantifier>Existential</quantifier>
 </time>
 <operation>Discharge</operation>
</commitment>
<commitment>
 <commitmentId>CL19</commitmentId>
 <debtor>DInL</debtor>
 <creditor>GML</creditor>
 <proposition>TI</proposition>
 <time>
  <from>Q1_B</from>
  <to>Q1_2</to>
  <quantifier>Existential</quantifier>
 </time>
 <operation>Discharge</operation>
</commitment>
<commitment>
 <commitmentId>CL20</commitmentId>
 <debtor>DInL</debtor>
 <creditor>GML</creditor>
 <proposition>TI</proposition>
```

```
 <time>
  <from>Q1_2</from>
  <to>Q2_3</to>
  <quantifier>Existential</quantifier>
 </time>
 <operation>Discharge</operation>
</commitment>
<commitment>
 <commitmentId>CL21</commitmentId>
 <debtor>DInL</debtor>
 <creditor>GML</creditor>
 <proposition>TI</proposition>
 <time>
  <from>Q2_3</from>
  <to>Q3_4</to>
  <quantifier>Existential</quantifier>
 </time>
 <operation>Discharge</operation>
</commitment>
<commitment>
 <commitmentId>CL22</commitmentId>
 <debtor>DInL</debtor>
 <creditor>GML</creditor>
 <proposition>TI</proposition>
 <time>
  <from>Q3_4</from>
  <to>Q4_E</to>
  <quantifier>Existential</quantifier>
 </time>
 <operation>Discharge</operation>
</commitment>
<commitment>
```

```
 <commitmentId>CS1</commitmentId>
 <debtor>EPrS1</debtor>
 <creditor>DPrL</creditor>
 <proposition>Produce</proposition>
 <time>
  <from>Q2_3</from>
  <to>Q3_4</to>
  <quantifier>Universal</quantifier>
 </time>
 <operation>Discharge</operation>
</commitment>
<commitment>
 <commitmentId>CS2</commitmentId>
 <debtor>EPrS1</debtor>
 <creditor>DPrL</creditor>
 <proposition>Produce</proposition>
 <time>
  <from>Q3_4</from>
  <to>Q4_E</to>
  <quantifier>Universal</quantifier>
 </time>
 <operation>Discharge</operation>
</commitment>
<commitment>
 <commitmentId>CS3</commitmentId>
 <debtor>EPrS2</debtor>
 <creditor>DPrL</creditor>
 <proposition>Produce</proposition>
 <time>
  <from>Q2_3</from>
  <to>Q3_4</to>
  <quantifier>Universal</quantifier>
```

```
    </time>
    <operation>Discharge</operation>
   </commitment>
   <commitment>
    <commitmentId>CS4</commitmentId>
    <debtor>EPrS2</debtor>
    <creditor>DPrL</creditor>
    <proposition>Produce</proposition>
    <time>
     <from>Q3_4</from>
     <to>Q4_E</to>
     <quantifier>Universal</quantifier>
    </time>
    <operation>Discharge</operation>
   </commitment>
  </commitments>

  <constraintsOnProposition>
   <constOnProp prop="Sell" req="Produce" relation="para"></constOnProp>
  </constraintsOnProposition>

</protocol>
```

REFERENCES

1. Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Englewood Cliffs, NJ, 2nd edition, 2003.

2. Scott A. DeLoach. Multiagent systems engineering of organization-based multiagent systems. In *SELMAS '05: Proceedings of the fourth international workshop on Software engineering for large-scale multi-agent systems*, pages 1–7, New York, NY, USA, 2005. ACM Press.

3. Gerard J. Holzmann. *Design and Validation of Computer Protocols*. Prentice Hall International, Inc., AT&T Bell Laboratories, Murray Hill, NJ, USA, 1991.

4. Pınar Yolum and Munindar P. Singh. Reasoning about commitments in the event calculus: An approach for specifying and executing protocols. *Annals of Mathematics and Artificial Intelligence*, 42(1-3):227–253, 2004.

5. Pınar Yolum. Towards design tools for protocol development. In *Proceedings of the 4th International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pages 99–105. ACM Press, July 2005.

6. Munindar P. Singh. Multiagent systems as spheres of commitment. In *International Conference on Multiagent Systems (ICMAS) Workshop on Norms, Obligations, and Conventions*, pages 312–326, 1996.

7. N. Fornara and M. Colombetti. Operational specification of a commitment-based agent communication language. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2002), Part II*, pages 535–542, 2002.

8. A. Artikis, J. Pitt, and M. Sergot. Animated specifications of computational societies. In C. Castelfranchi and L. Johnson, editors, *Proceedings of Conference on*

Autonomous Agents and Multi-Agent Systems (AAMAS), pages 1053–1062, 2002.

9. C. Castelfranchi. Commitments: From individual intentions to groups and organizations. In *Proceedings of the Int. Conf. on Multi-Agent Systems (ICMAS95)*, pages 41–48, 1995.

10. Munindar P. Singh. An ontology for commitments in multiagent systems. *Artif. Intell. Law*, 7(1):97–113, 1999.

11. Winfried Karl Grassmann and Jean-Paul Tremblay. *Logic and discrete mathematics: a computer science perspective*. Prentice Hall Press, Upper Saddle River, NJ, USA, 1996.

12. Ashok U. Mallya, Pınar Yolum, and Munindar P. Singh. Resolving commitments among autonomous agents. In Marc-Philippe Huget and Frank Dignum, editors, *Proceedings of the AAMAS Workshop on Agent Communication Languages and Conversation Policies, LNAI 2922*, pages 166–182. Springer Verlag, 2003.

13. J. F. Allen and H. A. Kautz. A model of naive temporal reasoning. In J. R. Hobbs and R. C. Moore, editors, *Formal Theories of the Commonsense World*, pages 251–268. Ablex, Norwood, NJ, 1985.

14. John Langshaw Austin. *How to Do Things with Words*. Oxford University Press, London, UK, 1962.

15. Amit K. Chopra and Munindar P. Singh. Contextualizing commitment protocol. In *AAMAS '06: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 1345–1352, New York, NY, USA, 2006. ACM Press.

16. Virginia Dignum, Javier Vázquez-Salceda, and Frank Dignum. Omni: Introducing social structure, norms and ontologies into agent organizations. In *AAMAS Workshop on Programming Multiagent Systems*, pages 181–198, 2004.

17. Marco Alberti, Federico Chesani, Marco Gavanelli, Evelina Lamma, Paola Mello, and Paolo Torroni. Compliance verification of agent interaction: a logic-based software tool. Number DEIS-LIA-04-003, 2005. LIA Series no. 71.