Formal Specification and Automatic Verification of Multi-Agent Conditional Commitments and their Applications

Warda El Kholy

A Thesis
In the Department
of
Concordia Institute for Information Systems Engineering

Presented in Partial Fulfillment of the Requirements
For the Degree of
Doctor of Philosophy (Information and Systems Engineering) at
Concordia University
Montréal, Québéc, Canada

April 2016

CONCORDIA UNIVERSITY
SCHOOL OF GRADUATE STUDIES

This is to certify that the thesis prepared

By: Mrs. Warda El Kholy

Entitled: Formal Specification and Automatic Verification of Multi-Agent Conditional Commitments and their Applications

and submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy (Information and Systems Engineering)

complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

_____ Chair
Dr. Anjali Agarwal

_____ External Examiner
Dr. Hanifa Boucheneb

_____ External to Program
Dr. Olga Ormandjieva

_____ Examiner
Dr. Abdessamad Ben Hamza

_____ Examiner
Dr. Benjamin C. M. Fung

_____ Thesis Supervisor
Dr. Jamal Bentahar

_____ Thesis Co-Supervisor
Dr. Rachida Dssouli

Approved by _____
Chair of Department or Graduate Program Director

_____
Dr. Amir Asif
Dean of Faculty of Engineering and Computer Science

Date _____ 2016

Abstract

Formal Specification and Automatic Verification of Multi-Agent Conditional Commitments and their Applications

Warda El Kholy, Ph.D.
Concordia University, 2016

Modeling agent communication using social commitments in the form of obligatory contracts among intelligent agents in a multi-agent system (MAS) provides a quintessential basis for capturing flexible and declarative interactions and helps in addressing the challenge of ensuring compliance with specifications. However, on the one hand, social commitments exclusively are not able to model agent communication actions, the cornerstone of the fundamental agent communication theory, namely *speech act theory*. These actions provide mechanisms for dynamic interactions and enable designers to track the evolution of active commitments. On the other hand, the designers of the system cannot guarantee the emergence of expected behaviors, such as self-contained intelligent agent complies with its protocols and honors its activated commitments. Moreover, the designers might still wish to develop effective and scalable algorithms to tackle the problem of model checking complex interactions modeled by conditional commitments and conditional commitment actions and regulated by commitment-based protocols at design time. Conditional commitments are a natural and universal frame of social commitments and cope with business conditional contracts. This dissertation is in principle about addressing two open challenging issues: 1) formally defining computationally grounded semantics for agent communication messages in terms of conditional commitments and associated actions (fulfill, cancel, release, assign and delegate), which is yet to be studied; and 2) developing a symbolic algorithm dedicated to tackle the raised model checking problem and to ensure the development of correct systems.

In this dissertation, we start with distinguishing between two types of conditional commitments: weak and strong. Weak conditional commitments are those that can be activated even if the antecedents will never be satisfied, while strong conditional commitments are those that can be solely activated when there is at least one possibility to satisfy their assigned antecedents. We develop a branching-time temporal logic called $\text{CTL}^{cc,\alpha}$ that extends computation tree logic (CTL) with new modalities for representing and reasoning

about the two types of conditional commitments and their actions using the formalism of interpreted systems. We present a set of valid properties, a set of reasoning rules, and a set of action postulates in order to explore the capabilities of CTL$^{cc,\alpha}$. Furthermore, we propose a new life cycle of conditional commitments. Having a new logic (CTL$^{cc,\alpha}$), we introduce a new symbolic algorithm to tackle the problem of its model checking. Instead of developing our algorithm from scratch, we extend the standard CTL model checking algorithm with symbolic algorithms needed for new modalities. We also investigate important theoretical results (soundness and termination) of the algorithm. Given that, we completely implement our algorithm and then assemble it on top of the symbolic model checker MCMAS, developed to automatically and directly test MAS specifications. The resulting symbolic model checker is so-called MCMAS+. We extend MCMAS's input modeling and encoding language called ISPL with shared and unshared variables needed for agent interactions and with the syntactic grammar of new modalities to produce a new one called ISPL+. We also extend the MCMAS's graphical user interface to display verified models to reduce inefficient and labor-intensive processes performed by the designers.

To evaluate the performance of the developed algorithm, we analyze its time and space computational complexity. The computed time and space complexity are P-complete for explicit models and PSPACE-complete for concurrent programs. Such results are positive because model checking CTL$^{cc,\alpha}$ has the same time and space complexity of model checking CTL although CTL$^{cc,\alpha}$ extends CTL. Therefore, CTL$^{cc,\alpha}$ balances between expressive power and verification efficiency. Regarding the feasibility aspect, we apply our approach in three different application domains: business interaction protocols, health care processes, and web service compositions. The MAS paradigm is successfully employed in these domains wherein a component is represented, implemented and enacted by an agent. The proposed approach improved the employed MAS paradigm by formally modeling and automatically verifying interactions among participating agents so that the bad behaviors can be detected and then eliminated or repaired at design time and the confidence on the safety, efficiency and robustness is increased. We conduct extensive experiments to evaluate the computational performance and scalability of MCMAS+ using very large case studies. The obtained results strongly confirm the theoretical findings and make MCMAS+ practical. We finally compare our approach to other available approaches and show that it outperforms such approaches in terms of execution time, memory usage and number of considered intelligent agents.

Acknowledgments

First of all, I would like to cordially thank my dissertation supervisor Dr. Jamal Bentahar for his invaluable support and patience, for his precious and never lacking enthusiasm for research, and for his continuous assistance in preparing, writing papers and this dissertation. I want to thank him for having so strongly believed in me and provided me with insights which helped me solve many of the challenging issues I encountered in my research. I would like to express my gratitude to my co-supervisor Dr. Rachida Dssouli whose expertise, understanding and patience, added considerably to my graduate experience. I want to warmly thank her for highly stimulating discussions that have led to the discovery of many results in this thesis. Moreover, I would like to acknowledge the Ministry of Higher Education, Egypt and my supervisors for respectively supporting and assisting me financially during my Ph.D., which gave me opportunities to complete this dissertation.

A very special thanks goes out to Dr. Hongyang Qu, University of Sheffield, Automatic Control and Systems Engineering Department, UK, for his extremely appreciated collaboration that allowed me to implement the symbolic model checker presented in this thesis. I wish to gratefully thank all members in my Faculty of Computers and Information, Menoufia University, Egypt. I also would like to thank all members in the Bureau of Cultural and Educational Affairs of Egypt in Montréal, Canada for their kindly help during my Ph.D. study.

My years as a Ph.D. student would not have been as much fun without my friends and colleagues with whom I shared the laboratory and unforgettable moments. From the administrative staff, Silvie Pasquarelli, Lilia Pernatozzi, Mireille Wahba and Rima Baroudi are thanked for their care and attention. A very special thanks goes out to my father (Ibrahim) and my mother (Hamedia) for the support they provided me through my entire life and for their endless patience. Without any doubt, a very special thanks goes out to my husband (Mohamed) and my children (Abdallah, Omar and Maryam) for being so patient with me, without their love, encouragement and editing assistance, I would not have finished this thesis; I love them so much.

To my parents and to my family,
with eternally love and gratitude

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter covers:

- The context of research and motivations.
- The challenges, research questions, contributions, and methodology of the dissertation.
- The dissertation organization.

1.1 Context of research

The paradigm of multi-agent systems (MASs) has received a growing attention. Gerhard Weiss [96] presented two reasons, which primarily play the driving forces behind the increase usage of MASs in recent years. The first reason is that MASs are successfully employed in several computer science applications ranging from simple to complex intelligent systems with stringent demands, such as processing huge amounts of data, which is distributed in geographically distinct locations. The second reason is that MASs provide a natural abstraction for "developing and analyzing models and theories", which can work effectively in solving complex human tasks, such as planning strategies, decision making, coordination, cooperation, and negotiation. MASs can be defined as "*systems in which several interacting, intelligent agents pursue some set of goals or perform some set of tasks*" [96]. From the definition, a key feature of any MAS is the ability of agents to interact with one another as well as their environments.

1.1.1 Agent properties

As shown in Figure 1[1], an agent is a persistent computational entity (e.g., a software program or a robot), which can perceive and act on its defined environment using its sensors and effectors through three phases: perception, decision and action [96, 100, 101]. A soft-

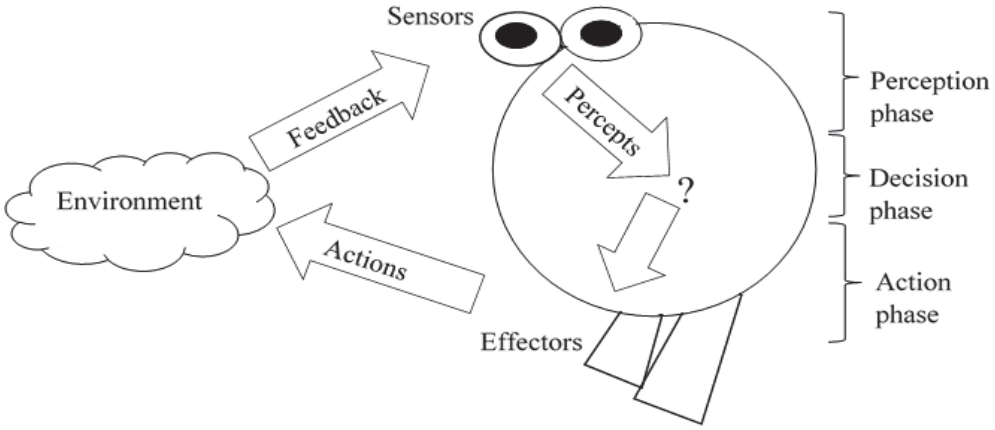

Figure 1: Agent interaction with its environment

ware agent can be autonomous and self-contained, meaning that the agent is free to interact with other agents as it delights, but acts according to the rationale of its local and social goals. This is because the agent has a full control on its internal states and own behaviors without a direct intervention of other agents or humans. In addition to the autonomy property, a software agent can satisfy the following properties [101]:

1. *Reactivity*: an agent is capable of responding to external changes within its environment.
2. *Pro-activity*: an agent is capable of behaving with respect to the rationale of its state of affairs, plans and goals.
3. *Social ability*: an agent is capable of interacting with other agents.

When an agent captures the capabilities of reactivity, pro-activity and social ability, it is so-called *intelligent agent* [101]. The intelligence property enables an agent to flexibly and rationally operate in various environmental circumstances. Since modern computing platforms and computer information systems are heterogeneous by nature, agents should satisfy the heterogeneity property in order to extend the range of practical MAS applications. The heterogeneity property means that agents are designed and implemented in

[1]This figure is adapted from `http://www.csc.liv.ac.uk/~mjw/pubs/imas/`.

different ways by different designers (i.e., agents are following different and probably opposite goals, desires, and intentions). On the other hand, homogeneous agents are those agents that have the same capabilities as they are designed in an identical way.

1.1.2 Agent communication

Since computer information systems are no longer stand-alone systems, it should be no surprise to anyone that interaction among autonomous and possibly heterogeneous intelligent agents implementing and enacting these systems is perhaps the best guideline to build effective MASs. In our understanding and thinking about the construction of effective MASs, a key pattern of interaction is goal-oriented coordination performed either in cooperative or in competitive business scenarios. That is, interacting intelligent agents might be affected by other agents in pursuing their goals. For example, in the case of cooperation scenarios, multiple agents interact with each other in order to assemble their capabilities to achieve as a team "what the individuals cannot" [96], due to the lack of resources, capabilities, or knowledge. With respect to competition scenarios, multiple agents attempt to bring "what only some of them can have" [96], i.e., the success of one agent entails the failure of other agents. The long-term goal of MASs is to develop mechanisms, which enable agents to interact and understand interaction with each other. Such goal raises a number of challenging issues, which are wholly centered around the elementary question of how intelligent agents interact with each other to successfully satisfy their design goals [96]. The following are some of these challenging issues: 1) what communication languages and interaction protocols to use?; 2) how to formally model the interactions among agents?; and 3) how to make sure that the interactions are correctly specified? Broadly speaking, an interaction protocol is needed to regulate and coordinate interaction among participants within conversations. An agent communication language (ACL) is a commonly understood artificial language, which enables intelligent agents to *talk* with each other and decide what proper information to exchange or right action to perform in order to achieve their goals. The research community of agent communication defined a suitable set of performative communicative acts to capture useful interactions among agents in a declarative form [98, 100].

Example 1.1.1. *An informative utterance such as "the shipment will be delivered on Monday" can be treated in a declarative form as "the seller informs the buyer that the shipment will be delivered on Monday".*

This dissertation focuses on a special communicative act called *commit*. In the literature of agent communication, the social commitment has two forms: unconditional and conditional [17, 49, 79]. Informally, unconditional commitment is a form of contractual obligation made by one intelligent agent, called debtor, as a result of messages exchanged or actions taken, and is directed towards another intelligent agent, called creditor, to bring about a certain fact (in what's also called a commitment consequence).

Example 1.1.2. *The seller unconditionally commits to the buyer to ship the requested goods.*

The basic idea of conditional commitments, as in most business and concrete applications, is that the debtor agent can merely commit towards the creditor agent to bring about consequences when specific antecedents are met.

Example 1.1.3. *The seller commits to the buyer to ship the requested goods if the buyer sends the agreed payment.*

Since conditional commitments are a natural and universal frame of social commitments and cope with business conditional contracts, our work and more recent research work [81, 26, 9, 59] consider conditional commitments as a first class and treat unconditional commitments as a special case when the antecedents are always true (i.e., when the agreed payment is sent). Now, why we elect social commitment-based approaches and symbolic model checking techniques.

1.2 Motivations

1.2.1 Social commitment-based approaches

The literature of agent communication is classified into two main categories: Mental approaches and social approaches. Both approaches have stressed the semantic part of agent communication by capturing pre- and post-conditions needed to exchange messages among agents. However, we adopt social commitment-based approaches employed successfully in social approaches to define semantics of ACL messages for the following reasons:

1. They provide a powerful engineering tool to represent, model, and reason about the content of interactions among pairs of social, autonomous, and heterogeneous agents [79, 9, 49, 55].

2. They provide social semantics which only defines public aspects of ACL messages exchanged among agents from high-level abstractions rather than reasoning about agents' mental states (e.g., desires and intentions). In this semantics, all states of MASs holding social facts such as commitments are global and accessible so that agents' behaviors can be easily monitored and verified [23, 85, 86, 93, 55].
3. They adopt the autonomy and flexibility of the interacting agents. The autonomy aspect is satisfied in a natural way since each agent is expected to only achieve its commitments. Since an agent can elect what is best for satisfying its goals, the commitment-based approaches support flexibility as long as the behavior is correct at the business interaction level [28, 97, 106, 55].
4. They provide a robust way to characterize the degrees of autonomy and interdependency of interacting agents without getting bogged down in low-level details [28, 55]. When the interacting agents are completely autonomous, we will not have an effective system of agents. On the other hand, when there is no interdependency, the interacting agents will be approximately useless; instead autonomous agents must be able to cooperate and compete with each other.

Furthermore, the theory of social commitments has been successfully applied to other research areas such as modeling business processes [36, 89, 90], developing web-based MASs [94], developing agent-based web services and their communities [47], specifying commitment-based protocols [105, 106, 34, 67], and specifying business protocols [36, 48]. Another important aspect of social commitments is that commitments can be manipulated through a set of commitment actions. Such manipulations precisely provide mechanisms for dynamic interactions and help designers track the evolution of active commitment states. They also provide a principle way to define commitment life cycles. Such actions are typically classified into two types: duplex actions (fulfill, cancel, and release), which have two parties, and triplex actions (assign and delegate), which include three parties [79, 94, 49].

Because software intelligent agents are autonomous, they can join, and leave systems at any time as well as they might not honor the received requests or be indisposed to supply services in the required time. On the one hand, the challenge of automatically detecting and then fixing such undesirable agents' behaviors according to preset specifications is an arduous issue to tackle in the mental semantic approaches. This is because we need an external or observer agent to access and read the mental or internal states of other agents,

which is not feasible. Moreover, mental semantics is intended to specify the informational properties of a system of agents. On the other hand, while social semantics of agent communication messages can effectively help tackle the challenge of insuring agents compliance with specifications [15, 53], the designers of the system cannot guarantee that an agent complies with its protocols and its social commitments as it is supposed to or at least an agent doesn't want to violate its commitments. Therefore, model checking techniques are an urgent requirement.

1.2.2 Symbolic model checking techniques

In a nutshell, a model checking is a formal and fully automatic verification technique performed at design time [32]. It is one of the most promising verification techniques that can help designers automatically detect and eliminate or repair undesirable agents behaviors. It specifically proves or disproves that a model (e.g., MAS) will satisfy given desirable properties. State explosion[2] is a phenomenon often encountered when using model checking based on automata-theoretic techniques [32]. One solution to alleviate this problem is to use symbolic model checking based on the de-facto and standard data structures called *ordered binary decision diagrams* (OBDDs). OBDDs are efficiently representing and manipulating the Boolean formulae encoding the system models symbolically. OBDDs are used in leading toolkits (e.g., NuSMV [29]) as they enjoy substantial features such as canonical representation and polynomial time transformations. We precisely adopt symbolic techniques because their algorithms are applied to Boolean functions not to Kripke structures [32], which therefore need less memory than automata-based techniques. In practice, space requirements for Boolean functions are exponentially smaller than for explicit Kripke structure representations. Moreover, symbolic techniques have been recently proven to be efficient techniques to automatically verify: 1) epistemic properties expressed using logics of knowledge [64]; 2) social communication properties expressed using logics of unconditional commitments [53, 15, 52]; and 3) the correctness of activity diagrams against certain properties [56].

[2]The state explosion problem means that the number of global states in a model grows exponentially with the number of variables, or concurrent components, constituting the modeled system.

1.3 Challenges, research questions and contributions

Previous approaches have considered the formal semantics of ACL messages in terms of social commitments by means of logics of actions or temporal logics. The approaches over the last 20 years have been reviewed using a set of evaluation criteria in [55]. From the findings of this survey paper [55], the approaches that use logics of actions (also called executable action languages) model social commitments as predicates [106, 97] or fluents [23, 25, 37]. Such modeling styles waive formal semantics and concrete meaning of commitments, which makes their verification very abstract. Artikis and Pitt in [6] argued that although the abductive event calculus planner [106] and causal calculator [25] that enact and execute commitment-based protocols[3] specified in the executable action languages called event calculus, causal logic C^+ and modular action description (an extension of C^+ to compose business protocols [37]) can be employed at run time, they can become inefficient when considering large MASs. Of course, executable action languages are very easily and efficiently implemented.

On the other hand, the approaches that adopt temporal logics model social commitments as temporal modalities and then use these modalities to extend: 1) linear temporal logic (LTL) [76, 81]; 2) computation tree logic (CTL) [15, 49, 51, 52, 79]; or 3) full computation tree logic (CTL*) [17, 16, 50, 54]. While developing such modalities is far from being easy, we adopt this approach as it allows us to: 1) have a standard form for representing and reasoning about social commitments; and 2) develop dedicated and implementable model checking algorithms for commitments and commitment actions. By deeply investigating the semantic models of conditional commitments introduced in the approaches that model social commitments as temporal modalities, we found in [41] that there are very few approaches [50, 54, 76, 81], which have tried to define a formal semantics for conditional commitments and such a semantics has strong limitations. According to our study in [41], the first limitation of conditional commitment semantic models results from using the linear temporal operator [76] and the implication operator [50, 54] to define the semantics of conditional commitments on top of the semantics of unconditional commitments; instead of developing a new modality for conditional commitments. Technically, this modeling inherits the fundamental limitation of these operators, stating that a conditional commitment could become active without satisfying its antecedent. Singh in [81] introduced an extended version of LTL with a new modality to represent and reason about

[3] These protocols are special kind of multi-agent interaction protocols specified as a set of commitments and commitment actions, which captures the meaning of protocols' messages.

conditional commitments in order to avoid using the strict implication operator introduced in [79, 104] and to solve the limitation of the implication operator introduced in [50, 54]. His semantics of conditional commitments is interpreted using Segerberg's approach of neighborhood semantics [81], which in turn maps "each world into a set of sets of worlds". The second limitation is that model checking such a semantics is still an open problem as it is not based on Kripke structures nor accessibility relations needed for defining semantics of standard modal operators. The third limitation is that Singh's semantics doesn't consider the possibility that antecedents of commitments could be always false. This possibility is indispensably needed for agents to adjust their interactive actions so as to take advantages of opportunities or to handle exceptions that arise during interaction. Thus, our initial research question is:

Question 1.3.1. *How to distinguish between different types of conditional commitments?*

We classified conditional commitment into two different but related subcategories: weak and strong [41, 39]. Indeed, weak conditional commitments capture the current flexibility of conditional commitment-based approaches, while strong conditional commitments capture real situations that weak conditional commitments cannot suitably model. At this point, the direct question is:

Question 1.3.2. *How to define a formal semantics of weak and strong conditional commitments?*

To address this question, we introduced two new temporal modalities to represent and model weak and strong conditional commitments [41, 39]. Then, we semantically defined weak commitment modalities as those that can be activated even if the antecedents will never be satisfied and strong commitment modalities as those that can be solely activated when there is at least one possibility to satisfy their assigned antecedents [41, 39]. The formal semantics of actions that can directly apply to a conditional commitment modality are still a challenging issue. The reasonable question that we explore here is:

Question 1.3.3. *How to define a suitable semantics of common conditional commitment actions?*

To address this question, we introduced new temporal modalities to represent and model all weak and strong conditional commitment actions (duplex and triplex actions) and defined their suitable formal semantics [44]. By 'suitable', we mean that the new formal semantics of conditional commitment actions should: 1) be reasonable, intuitive, and sound; 2)

consider axiom rules introduced in the literature to define the operational semantics of unconditional commitment actions (see for examples [23, 106, 97]) as there is no operational semantics for conditional commitment actions; and 3) address the spurious paradox appeared in the formal semantics of fulfilling unconditional commitments [15, 53, 52]. This paradox in fact results from the counterintuitive assumption that "the unconditional commitment should be active when it comes time to its fulfillment". The following example clarifies the paradox:

Example 1.3.1. *Suppose a customer commits to give $500 to a merchant. As soon as that money is transferred to the merchant's account, the commitment is immediately fulfilled. By considering this assumption, the commitment to send $500 is still active, but it would be ridiculous to force the customer to send the money again.*

This assumption technically violates the following principle that is commonly accepted in the literature [106, 97, 23]: when a commitment is fulfilled, it should become no longer active, meaning that the fulfillment action results in a state where the active commitment is marked as resolved (or terminated). As mentioned earlier, there are different temporal logics in the literature of agent communication, thus the expected question is:

Question 1.3.4. *Which temporal logic we select to represent and reason about conditional commitments and their actions?*

Although LTL and CTL are incomparable expressiveness wise, the standard model checking algorithm for LTL is exponential in the length of the formula and linear in the size of the model [77] and for CTL it is linear in both the length of the formula and model [77]. Moreover, it is known as a fact that CTL* is more expressive than LTL and CTL, but its model checking algorithm has the same complexity as the LTL algorithm [77]. With these arguments, the CTL logic is a suitable language, especially there are several open model checkers that support this logic. However, CTL is missing the capabilities to model interactions and dynamic behaviors of intelligent agents. Following current approaches that address one of the limitations of CTL by adding unconditional commitment modality to model the interactions among agents, the first contribution which addresses the Questions 1.3.1, 1.3.2, 1.3.3, and 1.3.4 is as follows:

Contribution ①: we extended the standard CTL logic with new temporal modalities to represent and reason about weak and strong conditional commitments and their actions and then defined formal semantics for these modalities [41, 39, 44].

Given that, the direct question is:

Question 1.3.5. *How can we formally define properties and reasoning rules about the developed temporal logic?*

In fact, properties and reasoning rules explore the capabilities of the developed logic and agents are expected to respect them when they communicate. For example, one category of the valid properties captures the relationship between strong and weak commitments. Also, one motivation behind our reasoning rules is to capture common reasoning patterns that uniformly apply to weak and strong conditional commitments and their actions. By addressing this question, the second contribution is as follows:

Contribution ②: we presented a set of valid properties and a set of reasoning rules along with their formal proofs in [39] as well as a set of action postulates in [44].

In order to reduce and eliminate post-development costs and increase confidence on the safety, efficiency and robustness, two verification techniques at design time have been put forward to verify unconditional commitments and associated actions, and commitment-based protocols: direct and indirect verification techniques (see Figure 2). The idea of direct techniques is to develop algorithms needed to tackle the problem of model checking unconditional commitment and fulfillment action modalities at design time [15, 53]. The

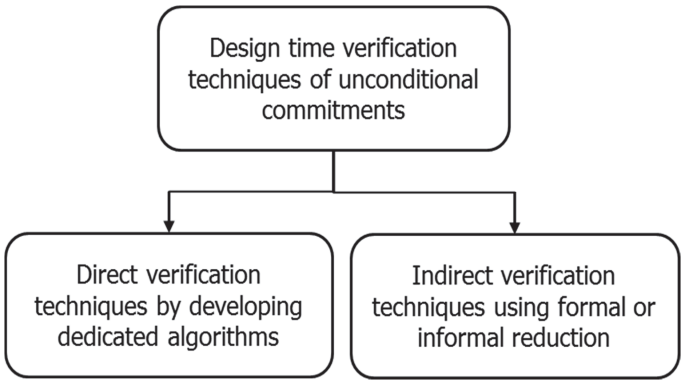

Figure 2: Current design time verification techniques of unconditional commitments.

indirect techniques are based on transforming the model checking problem into another one to be able to use existing model checkers. Such techniques are implemented by either formal [51, 52, 54] or informal [49] reduction methods. Based on our experience in [52, 54], we elect the direct technique to address the limitations of the indirect technique in terms of high memory consumption and limited scalability with regard to the number of considered agents. Here, the question is:

Question 1.3.6. *How can we design and implement a symbolic algorithm for the developed temporal logic?*

By addressing this question, the third contribution is as follows:

> **Contribution ③:** we developed a symbolic algorithm which invokes sub-algorithms needed to model check our new modalities [39, 44]. We also implemented our algorithm on top of the MCMAS symbolic model checker—developed to verify the correctness of MASs [64]—to produce a new symbolic model checker. The graphical user interface of our model checker is presented in [43].

To compute the necessary and sufficient resources for solving all problems' instances, including the hardest case, we need to consider the following question from the computation perspective:

Question 1.3.7. *What is the time and space complexity of the model checking problem of the developed temporal logic?*

By addressing this question, the fourth contribution is as follows:

> **Contribution ④:** we computed the time and space complexity of the model checking problem of the developed temporal logic. We found that this problem is P-complete for explicit models and PSPACE-complete for concurrent programs [44]. Such results mean that the cost of our algorithm is similar to the corresponding one of the standard CTL algorithm.

By the end, the expected question is:

Question 1.3.8. *How can we apply the commitment language and its model checker to different application domains?*

By addressing this question, the fifth contribution is as follows:

> **Contribution ⑤:** we successfully applied our language and tool to model check business protocols [44, 39, 43], health care processes [39], and Web service compositions [42, 40]. This contribution goes beyond the simple application to cover a new methodology of expressing specifications in terms of commitments and their actions and to model the system using refinements and compositions.

1.4 Methodology

We now describe the proposed methodology, which is summarized beside contributions in Table 1. At the beginning of this dissertation, we reviewed and evaluated relevant approaches that use executable action languages and CTL*, LTL and CTL logics to define the formal semantics of social commitments and associated actions capturing the meaning of ACL messages against a set of evaluation criteria. According to a joint publication in [55], we found that the current executable action languages and CTL* and LTL logical languages are unsuitable in terms of social commitment modelings and complexity efficiency, respectively. Therefore, we elected CTL to be our temporal logic. Then, we proceeded to review and evaluate the current CTL-based approaches devoted to define a formal semantics for ACL messages using our crucial criteria. This review is the core of Chapter 2. Unfortunately, there is no formal semantics for conditional commitments and their actions and most of these approaches considered only unconditional commitments. Before starting to address this gap, we studied a set of motivating examples, which revealed the importance of distinguishing between weak and strong conditional commitments. By identifying this importance, we started by enriching CTL with novel temporal modalities to represent and reason about conditional commitments and their duplex and triplex actions. The resulting temporal logic is so-called CTL$^{cc,\alpha}$. Our new formal semantics has the advantage of having a direct and concrete interpretation into computational models, which yields the quality of being computationally grounded. To explore the capabilities of CTL$^{cc,\alpha}$, we presented a set of reasonable and valid properties of CTL$^{cc,\alpha}$, which capture the relationship between weak and strong commitments and their fulfilments as well as the relationship between conditional and unconditional commitments. We also presented a set of reasoning rules and their proofs, which capture the characteristics of commitments and their actions. These rules also provide a generic way to deal with exceptions and enable us to define the life cycle of conditional commitments. Furthermore, we presented a set of action postulates in the form of business patterns. With these theoretical properties, we can underline that our semantics basically satisfies the main functionalities of neighborhood semantics [81] and captures all operational semantic rules that are widely discussed in the literature [106, 97, 23].

An important aspect in the proposed approach is that we introduced an extended version of the interpreted system formalism proposed in [15, 52] to account for agent communication with the reachability condition in the definition of the accessibility relation. Indeed, our

Table 1: Our methodology steps

Step	Description	Input	Output and contribution
1	Reviewing and evaluating	Current approaches and a set of evaluation criteria	−Advantages and limitations
2	Selecting a suitable temporal logic	Expressiveness and complexity of model checking LTL, CTL, and CTL*	−CTL (balance between expressiveness and complexity)
3	Distinguishing different types of conditional commitment	Concept of conditional commitments and a set of motivating examples	−Weak and strong commitments
4	Extending CTL with temporal modalities	CTL and new modalities and their semantic rules	−CTL$^{cc,\alpha}$ −Contribution ①
5	Reasoning about CTL$^{cc,\alpha}$	CTL$^{cc,\alpha}$	−A set of reasoning rules and a set of action postulates −Contribution ②
6	Defining CTL$^{cc,\alpha}$ properties	CTL$^{cc,\alpha}$	−A set of valid properties −Contribution ②
7	Developing a symbolic model checking algorithm for CTL$^{cc,\alpha}$	CTL$^{cc,\alpha}$ and CTL model checking algorithm	−A symbolic algorithm dedicated to CTL$^{cc,\alpha}$ −Contribution ③
8	Implementing the developed algorithm	MCMAS and its user interface along with our symbolic algorithm	−ISPL+, MCMAS+ and new user interface with a new capability to render models −Contribution ③
9	Computing complexity	Our model checking algorithm	−Time and space complexity −Contribution ④
10	Applying our language and tool	CTL$^{cc,\alpha}$, MCMAS+, ISPL+ and specifications of 3 application domains	−Modeling, expressing, encoding, experimenting and correcting each model application −Contribution ⑤

extension enables us to address the fulfillment paradox appeared in [15, 53, 52]. As a matter of fact, this formalism provides a mainstream framework for modeling, reasoning and systematically exploring fundamental classes of MASs, such as the class of synchronous systems [57].

Although there are two model checker tools [15, 53] recently developed to verify automatically unconditional commitments, we could not use them to verify conditional commitments. This is because, on the one hand, our temporal model and our semantics of conditional commitments and their actions are entirely different. On the other hand, such tools only support the verification of fulfilling unconditional commitments. For these technical reasons, we developed new algorithms dedicated to the new modalities. These algorithms

in fact extend the standard CTL algorithm [32]. We fully implemented these algorithms on top of the MCMAS model checker [64], producing the MCMAS+ tool and its input language called ISPL+. For better usability and ease-of-use, we extended the graphical user interface of MCMAS to help prospective users and designers edit and encode design models, and render the labeled transition systems of models.

Thereafter, we analyzed the performance of our symbolic algorithm in terms of time and space complexity. Finally, to check the effectiveness, scalability, and applicability of our approach, we reported extensive experimental results of verifying several interesting case studies, taken from different application domains, given desirable properties. We also compared the obtained results with other approaches (if any). Such results strongly confirm the theoretical findings.

1.5 Dissertation organization

According to the proposed methodology, the rest of the dissertation is divided into three parts. We also link research questions with chapters that address them.
Part I is about the state of the art:

- **Chapter 2** presents the syntax and semantics of CTL. It briefly presents the proposed criteria for reviewing and evaluating current approaches that use CTL to define a semantics for ACL messages in terms of social commitments and related actions and to specify commitment-based protocols. The chapter reviews how commitments and actions are modeled along with their formal semantics (if any) and different verification techniques proposed to verify these protocols. The main outcomes of the chapter are to highlight advantages and limitations in the current approaches. Such limitations are addressed in the dissertation.

Part II presents our contributions (from 1 to 4) and consists of three chapters:

- **Chapter 3** gives an overview about the proposed approach along with a set of motivating examples. These examples show the need for distinguishing between weak and strong conditional commitments. The chapter briefly summarizes the formalism of interpreted systems and our extended version of interpreted systems and then presents the syntax and semantics of the developed CTL$^{cc,\alpha}$ logic. A set of properties, a set of reasoning rules, a set of action postulates and the conditional commitment life cycle are also presented. The main contribution of the chapter is to address the research Questions from 1.3.1 to 1.3.5.

– **Chapter 4** capitalizes on our new symbolic algorithm dedicated to CTL$^{cc,\alpha}$. The chapter also presents theoretical results of the developed algorithm. The chapter then moves to present the full implementation of our algorithm on top of the MCMAS symbolic model checker. Furthermore, our implementation extends both the input language and the graphical user interface of MCMAS to parse the CTL$^{cc,\alpha}$ syntax and help designers design, encode, verify and view models. The main contribution of the chapter is to address the research Question 1.3.6.

– **Chapter 5** analyzes theoretically the computational complexity of our algorithm in terms of running time and memory usage. The main contribution of the chapter is to address the research Question 1.3.7.

Part III presents the application domains of our approach (contribution 5) and concludes the dissertation. It consists of two chapters:

– **Chapter 6** focuses on the application domains of our commitment language and its model checker tool. Specifically, it considered three application domains: business interaction protocols, health care processes, and Web service compositions. The main contribution of the chapter is to address the research Question 1.3.8.

– **Chapter 7** summarizes the obtained results in this dissertation, presents open issues and sketches possible extension of this work.

Chapter 2

Background and Literature Review

This chapter covers:

- The syntax and semantics of CTL.
- The evaluation criteria.
- The review and evaluation of current CTL-based approaches.

2.1 Background about CTL

Clarke and Emerson in the early of 1980s introduced a branching time logic termed a computation tree logic (shortly, CTL) for two purposes: specification and verification [30]. The underlying modeling time in CTL is assumed to have a tree-like structure, which refers to the fact that each moment in time might divide into different possible paths in the future. This branching structure enables CTL to capture the nondeterminism of software programs in which any one of the paths might be an actual path, which is realized. CTL requires that a path quantifier—either A ("along all paths" or inevitably) or E ("along at least one path", "there exists one path" or possibly)—is immediately followed by a single operator from the menu of usual temporal operators: G ("globally"), F ("sometime"), X ("next time") or U ("until"). All legal formulae in CTL are state formulae.

Definition 2.1.1. *The syntax of CTL formulae over the underlying set $\mathcal{AP} = \{p, q, \ldots\}$ of atomic propositions is given by the following BNF grammar [32]:*

$$\varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid EX\varphi \mid EG\varphi \mid E(\varphi\, U\, \varphi)$$

where $EX\varphi$ is read as "there exists a path such that at the next state of the path φ holds", $EG\varphi$ is read as "there exists a path such that φ holds globally along the path", and

$E(\varphi\, U\, \psi)$ is read as "there exists a path such that ψ eventually holds and φ continuously holds until then".

Because the above syntax includes a minimal set of operators, other operators can be introduced in this syntax as abbreviations. For example, the Boolean implication operator $\varphi \rightarrow \psi$ abbreviates $\neg\varphi \vee \psi$, constant true proposition $\top$ abbreviates $p \vee \neg p$, conjunction $\varphi \wedge \psi$ abbreviates $\neg(\neg\varphi \vee \neg\psi)$ and equivalence $\varphi \equiv \psi$ abbreviates $\varphi \rightarrow \psi \wedge \psi \rightarrow \varphi$. Moreover, the syntax of CTL can incorporate the following temporal operators as abbreviations: $AX\varphi$, $AG\varphi$, $A(\varphi\, U\, \psi)$, $EF\varphi$ and $AF\varphi$. These are read, respectively: "along all paths, in the next state φ holds", "along all paths, φ holds globally", "along all paths, φ holds until ψ holds", "there exists a path such that φ holds at some point in the future" and "along all paths, φ holds at some point in the future". The universal path quantifier A can be defined by the existential path quantifier and negation (e.g., $\neg AX\varphi \equiv EX\neg\varphi$). Since G and F are duals, an instance of De Morgan's law can be defined in CTL as follows: $\neg AF\varphi \equiv EG\neg\varphi$, and $\neg EF\varphi \equiv AG\neg\varphi$.

The semantics of CTL formulae is given in terms of a transition system $M = (S, T, V, I)$ where S is a nonempty set of states, $T \subseteq S \times S$ is a transition relation, $V : S \rightarrow 2^{\mathcal{AP}}$ is a valuation function, and $I \subseteq S$ is a set of initial states. The transition relation T models temporal transitions among states: given two states $s, s' \in S$, $(s, s') \in T$ means that s' is an immediate successor of s. Conventionally, every state should has a successor state, i.e., the transition relation is total.

The behaviour of the system model M is a set of computation paths. Each path π in M is an infinite sequence of states (i.e., $\pi = s_0, s_1, \ldots$) such that $(s_i, s_{i+1}) \in T$ for all $i \geq 0$. The i^{th} state in the path π is denoted by $\pi(i)$.

Definition 2.1.2. *The satisfiability relation denoted by $M, s \models \varphi$ means that the formula φ holds at state s in a system model M and is defined inductively as follows:*

- $M, s \models p$ *iff* $p \in V(s)$,
- $M, s \models \neg\varphi$ *iff* $M, s \not\models \varphi$,
- $M, s \models \varphi \vee \psi$ *iff* $M, s \models \varphi$ *or* $M, s \models \psi$,
- $M, s \models EX\varphi$ *iff* $\exists\pi$ *such that* $\pi(0) = s$ *and* $M, \pi(1) \models \varphi$,
- $M, s \models EG\varphi$ *iff* $\exists\pi$ *such that* $\pi(0) = s$ *and* $M, \pi(i) \models \varphi\, \forall i \geq 0$,
- $M, s \models E(\varphi\, U\, \psi)$ *iff* $\exists\pi$ *and for some* $k \geq 0$ *such that* $\pi(0) = s$ *we have* $M, \pi(k) \models \psi$ *and* $M, \pi(i) \models \varphi\, \forall 0 \leq i < k$.

The interpretations of atomic propositions, Boolean operators, and temporal operators are defined as usual (see for example [32]).

2.2 Evaluation criteria

In this section, we present nine evaluation criteria. In fact, the first four agent communication criteria were introduced by Singh [79] to get a well-defined semantics for ACL messages. Three evaluation cetraria called commitment-oriented criteria are introduced in [55]. In the dissertation, we augmented commitment-oriented criteria with two commitment action-oriented criteria in order to capture how social commitments and their actions are modeled and verified, and to check whether or not there exists a formal semantics for commitments and their actions. The criteria are:

1. **Formal**. The language should have a formal syntax and semantics to eliminate the possibility of ambiguity in the meaning of ACL messages and to allow agents to reason about them.
2. **Declarative**. The semantics should focus on what interactions and protocols are about to avoid over-constraining the interactions by specifying how interactions should be accomplished. Combing logical languages and reasoning techniques is a central aspect to meet the declarative criterion.
3. **Meaningful**. The semantics should capitalize on the content and meaning of messages, not on their representation as sequences of tokens to enable agents to deal better with exceptions and opportunities.
4. **Verifiable**. We can check if agents are behaving according to the given semantics and/or protocols.
5. **Commitment Modeling**. How commitment is formally modeled, for example as a predicate in the propositional logic or as a temporal modality.
6. **Action Modeling**. How commitment action is formally modeled, for example as proposition or as temporal modality. By knowing the way of modeling commitments and commitment actions, we can check whether those commitments and actions have a grounded and concrete semantics or not.
7. **Commitment Semantics**. The commitment along with its content should be formally and intuitively captured.

8. **Action Semantics**. The commitment action should be formally and intuitively captured.
9. **Verification Method**. What is the technique used to verify the correctness of commitments and associated actions along with commitment-based protocols against specifications (e.g., model checking). This criterion can help designers select a verification method based on their own needs.

2.3 Literature review

Our methodological review has two parts: 1) explain each approach and point out carefully how it meets or not the nine criteria introduced in the previous section; and 2) evaluate a verification technique introduced in this approach to verify a commitment-based protocol. The key point is to highlight advantages and limitations in existing approaches. Such limitations are addressed in the dissertation. We classify current approaches into: 1) pure CTL-based approaches; and 2) extended CTL-based approaches. The first class abstracts some of atomic propositions (assertions) to have the form of predicate variables with a certain name and certain arguments, one of these arguments is the subject. This abstraction is needed to represent commitment actions and evaluated them to truth or falsity. Therefore, it waives real semantics of commitment actions. The second class extends CTL with temporal modalities to represent and reason about commitments and commitment actions. In both classes, CTL modalities are the only possibility to reason about time.

2.3.1 Pure CTL-based approaches

The authors in [94] introduced an approach to check whether or not the behavior of an agent participating in web-based MASs complies with a commitment-based protocol. This approach considers only unconditional commitments where each commitment is abstractly modeled as a variable c of certain type to produce an abstract data type. Operations (actions) that can be performed on commitments are seen as methods that have c as an argument. They modeled "the operations as propositions" in CTL-formulae, for example:

Example 2.3.1. *Let $c = C(Mer, Cus, deliverGoods)$ be a variable that models a commitment from a merchant Mer to a customer Cus to deliver the requested goods. The commitment c is fulfilled by Mer when the proposition $Discharge(Mer, c)$ is true. The truth evaluation of the $Discharge(Mer, c)$ operation depends on the proposition $deliverGoods$.*

To temporally reason about commitment operations, they defined the following rule schema in the protocol specification using CTL modalities: $MetaProp ::= AG(Bool \rightarrow AF\ Act) \mid AG(Act \rightarrow Bool)$ where $Bool$ is a Boolean combination of actions (Act), commitments or domain-specific concepts. When it comes to agent communication criteria, the semantics is formal (since the specification of protocols is specified in CTL), declarative, meaningful (as the meaning of every message is expressed in terms of commitments) and verifiable. Their verification process is centred on the condition under which an observer agent participating in the protocol can check the satisfaction or violation of the active commitments made by other agents using a run time verification technique similar to a design time model checking technique. Technically, when the observer agent discovers an inappropriate execution, this means that the system doesn't satisfy the protocol. The authors considered the fish-market protocol regulating interaction between two agents as a case study without discussing any experimental results.

Limitations:

1. There is no formal semantics for commitments and their actions in the proposed modeling way.
2. The proposed verification method doesn't verify the correctness of the composition of interacting agents and is not effective when the system under consideration has a large state-space.

The authors introduced in [102] a set of commitment patterns to collectedly model agent interactions. In this approach, unconditional commitments are only considered and simply modeled as abstract data types. The commitment consequence is defined as a predicate with a vector of arguments in order to pass the required data values, for example:

Example 2.3.2. *The commitment* $C(Cus, Mer, NB, sendPayment(\overrightarrow{price, date}))$ *means that* Cus *commits to* Mer *in the context of the NetBill protocol to send the payment, which is a predicate with two arguments: the price and date of the good item.*

Each commitment pattern captures a certain scenario and combines between communication primitives and commitment operations. Moreover, each commitment pattern can be formalized as a CTL formula, for example:

$$\forall i, j, Pred, \vec{v}: \ AG[Withdraw(i, j, Pred(\vec{v})) \rightarrow AF[Cancel(i, C(i, j, G, Pred(\vec{v})))]]$$

The left side of this formula refers to the communication part, while the right side refers to the commitment operation part. The latter part captures the meaning of the communication part from high-level abstraction. The formula intuitively states that along all paths

globally when agent i performs the withdraw message containing a predicate $Pred(\vec{v})$, then along all paths there is a possibility that i cancels a commitment to bring about the same predicate towards agent j in the context G. It is clear that the the communication message *Withdraw* and commitment action *Cancel* are abstracted as predicate variables (or propositions). Moreover, the authors translated each commitment pattern into a statechart diagram and then merged multiple statechart diagrams into one statechart diagram to describe a state machine defining the behavior model of each agent. This behavior model is used to generate a CTL model. The authors improved their approach by introducing: 1) an algorithm, which transforms the statechart model of agent into CTL model; and 2) a complete library of commitment patterns [103]. They also showed how certain behavior models of agents are sound for a certain set of patterns using a theorem proving technique. From agent communication criteria, the semantics of this approach is formal, declarative, meaningful, and verifiable. The authors considered an e-commerce scenario between a customer, travel agent, airline agent, and hotel agent as a case study without discussing any experimental results.

Limitations:

1. The proof construction of theorem proving fails "to be of much help" even for the simplest logics [31].
2. With respect to commitment-oriented and commitment action-oriented cetraria, there is no formal semantics because commitment actions applied to abstract data types (commitments) are simply treated as propositions in CTL formulae.

The authors in [66, 68] introduced a set of predicates to define the operational semantics of unconditional commitment actions. For instance, the $Satisfied(c)$) predicate is used to define the semantics of discharging commitments as follows:

$$M, s \models Satisfied(c) \textit{ iff } \exists s_3 : s_3 \leq s \textit{ and } M, s_3 \models Discharge(i, c) \textit{ and},$$
$$(\exists s_1 : s_1 < s_3 \textit{ and } M, s_1 \models Create(i, c), \textit{ and } (\forall s_2 : s_1 \leq s_2 < s_3 \rightarrow M, s_2 \models Active(c)))$$

When a commitment is created, but not discharged, canceled, delegated, released, and assigned yet, the status of the commitment is active. The $Satisfied(c)$ predicate is satisfied in a model M at s iff the discharging commitment c (i.e., $Discharge(i, c)$) is true and this commitment has been created in the past and still active. That is, the performance of the discharge action is being essentially equal to the satisfaction of the proposition p as stated

clearly by the authors: "if p occurs, the discharge action is assumed to have happened". The authors also introduced two temporal quantifiers (existential and universal) to define deadlines of commitment consequences in the figure of time instants and intervals, while unconditional commitment itself is modeled as abstract data type, for example:

Example 2.3.3. *If the proposition p represents a price quote, then $[d_1, d_2]p$ means that p will be an offer for the period between d_1 and d_2. The commitment $C(Cus, Mer, [d_1, d_1 + 24hours]p)$ states that Cus can commit to Mer to send the price p, which is only valid for an entire day.*

Such a semantics meets formal, declarative, and meaningful criteria, but not the verifiable criterion. The authors considered the scenario of a travel agent, who plans to book an airline ticket, a rental car, and a hotel room as a motivating example without discussing any experimental results.

Limitation:

1. With respect to commitment-oriented and commitment action-oriented cetraria, there is no formal semantics because commitment actions are abstracted as propositions and commitments are modeled as abstract data types.

The authors introduced in [59] a tool called "Proton" to specify commitment-based protocols and their refinements. The refinement process is inspired by the notion of refining super-protocol by sub-protocol in software engineering as along as each computation allowed by the sub-protocol is also allowed by the super-protocol. For example, a protocol PayViaMM refines a protocol Pay since "PayViaMM makes a payment as Pay specifies". Proton declaratively specifies a protocol in terms of: 1) a set of agent roles; 2) guarded messages, which need to be true before sending the messages by the roles; and 3) the meaning of each message. This meaning is defined as a set of actions applied to the social states of roles. Such states hold propositions, which indeed specify the states of commitments. Moreover, Proton describes the syntax for a refinement mapping. The authors used model checking to check whether a protocol correctly refines another one with respect to a given mapping, which contains the essential elements for refining two protocols. Technically, the proposed verification technique depends on the Proton preprocessor, which reads the two protocols and mapping specifications and outputs both the encoding model accepted by the MCMAS tool [64] and CTL formulae expressing certain requirements, for example:

Example 2.3.4. *When an action's sub-guard is true, its super-guard must also be true. Proton generates CTL formulae expressing this requirement as follows: $AG(a.sub\text{-}guard \rightarrow a.super\text{-}guard)$.*

When all CTL formulae are satisfied in the generated model, the protocol refinement holds successfully. Finally, the authors reported the experimental results of verifying the refinement of 16 commitment-based protocols and the elapsed time for the refinement verification. The maximum number of agents participating in these protocols are 3 agents (in the NetBill3 protocol). From agent communication criteria, the proposed semantics is formal, declarative, meaningful, and verifiable. Conditional and unconditional commitments are modeled as objects with seven instances (or states). Such objects are abstractly mapped into domain variables in the model accepted by MCMAS. Moreover, the authors require to detach conditional commitments into unconditional ones when the antecedents are true to be able to apply actions on the resulting commitments.

Limitations:

1. It is clear that objects and their mapped variables cannot represent the concrete meaning of conditional commitments.
2. Actions applied to unconditional commitments are modeled as atomic propositions, not effective actions. For example, the operational semantics of the discharge action "occurs implicitly when the consequent becomes true".

The authors developed in [89, 90] a business model, which uses conditional and unconditional commitments and agent-oriented concepts (e.g., goals and tasks) to capture complex business scenarios among business partners incorporated in service engagements [89] and cross-organizational business processes [90]. The authors also developed a library of business patterns wherein each pattern is defined from high-level abstraction using the notion of social commitments with some attributes (e.g., *intent, motivation,* and *implementation*). The authors introduced two different verification methods. In the former method [89], they introduced a reasoning algorithm that takes as input a business model produced from a set of business patterns and business interactions formalized using the UML sequence diagrams and outputs a set of violated commitments. In the latter method [90], they used the symbolic NuSMV model checker [29] to verify whether a set of business interactions complies correctly with the defined business model. A business model pattern is formalized as a CTL formulae, for example:

Example 2.3.5. *When a commitment is detached in the current state, then in the next state it might be detached, satisfied, violated, or terminated. This pattern can be formalized as:* $AG(Detached \rightarrow AX(Detached \vee Satisfied \vee Violated \vee Terminated))$

A conditional commitment notation is modeled as an SMV module, which can be initiated as a simple variable in the main module. The authors considered the quote to cash

(QTC) business process as a case study, which is specified and modeled using 17 conditional commitments among 6 agents. From agent communication criteria, this semantics is formal, declarative, meaningful, and verifiable.

Limitations:

1. There is no formal semantics for conditional commitments as the commitment notations are simply modeled as SMV modules.
2. Formal semantics of conditional commitment actions is not considered.

Kafali et al. [61] developed a tool called $\mathcal{PROTOSS}$ to detect and predict possible privacy violations within online social networks (OSNs). The privacy agreements that exist among each user and the OSN operator and the relations among the users constitute the formal model. Such privacy agreements are represented as a set of commitments. The tool has a semantic reasoning component that makes use of an ontology, which is used to refine commitments. For instance, if the OSN operator commits to a user not to share her information, then via reasoning on the ontology, the system can infer that neither a person's location nor her pictures can be shared. Moreover, undefined privacy concerns can be discovered through the ontology. The following example shows a commitment representing a privacy agrement.

Example 2.3.6. $CC(operator, charlie, colleague(charlie, Ur), \neg shareLocation(charlie, Ur))$*, where the social network operator (*$operator$*) is the debtor, Charlie is the creditor, the antecedent* $colleague(charlie, Ur)$ *represents* $charlie$ *and some another user* Ur *are colleagues and the consequent* $\neg shareLocation(charlie, Ur)$ *represents that the location information of* $charlie$ *is not shared with* Ur*.*

The above example shows that the operator will be committed not to share Charlie's location information with his colleagues if Charlie declares an individual as his colleague. After these agreements are specified, then the OSN operator is asked to check if there are any privacy violations in the system model. Following the above example, if a colleague ends up seeing Charlie's location, this would yield a privacy violation. Kafali et al. represented privacy violations as commitment violations in the system model and employ model checking to detect such violations. In general, the model checking technique is used to check that the private behavior rules—which define the operational behavior of the OSN operator—comply with privacy agreement shared between a user and the OSN operator, such that the operator would act in a way that honors its agreement with the user. Specifically, the authors use NuSMV as the underlying model checker and model a commitment

as an SVM module in order to enable NuSMV to deal with commitments, in the same spirit of [90]. The commitment module defines the statuses of a commitment, which correspond to the commitment states. Since the focus is on the privacy violations, only four commitment states are implemented: Conditional, active, fulfilled, and violated. The life cycle implemented there allows a commitment to be fulfilled only if both the antecedent and consequent hold since if the consequent holds without the antecedent, there is still a privacy violation. Using the commitment module, domain variables of commitment types can be specified. With the help of $\mathcal{PROTOSS}$, which is implemented as the privacy checker for the OSN operator, privacy properties are expressed in CTL and then automatically checked. For example, the following property describes if Charlie and Linus are colleagues, the commitment introduced in Example 2.3.6 (say $c1$) is violated. The model checking then decides if this is true, thereby leading a privacy violation:

$$AG(colleague\_charlie\_linus \rightarrow AF\ c1.status = Violated)$$

The approach has been tested over privacy scenarios. The authors generated OSN models with varying size of users (from 3 to 20 users) such that all users are related to each other. They also reported the number of states, the memory used, and the time consumed. However, the time and memory consumption increase exponentially when the model grows.

Limitation: since Kafali et al.'s proposal follows the same methodology as Telang and Singh [90] when it comes to the representation of conditional commitments as SMV modules, the two proposals share the same evaluation and limitations with regard to our criteria.

Table 2 summarizes our results of reviewing the current pure CTL-based approaches. In the table, we use **For.**, **Dec.**, **Mea.**, **Ver.**, **Mod.**, **Sem.** and **Verific. Method** to respectively refer to formal, declarative, meaningful, verifiable, modeling, semantics and verification method criteria. We also use * to refer to the approach that considers only unconditional commitments.

2.3.2 Extended CTL-based approaches

In 2000s, Singh [79] introduced an approach to define the meaning of ACL messages by associating with each communicative act three reasonable claims. The first claim is called *objective claim*, stating that the debtor agent is committed to send something which is true. The *subjective claim* is the second one and means that the debtor agent believes what is

Table 2: Summary of pure CTL-based approaches

	Agent Communication Semantics				Commitment		Action		Verific. Method
	For.	Dec.	Mea.	Ver.	Mod.	Sem.	Mod.	Sem.	
Venkatraman & Singh* [94]	●	●	●	●	abstract data type	○	proposition	○	model checking -like
Xing & Singh * [102, 103]	●	●	●	●	abstract data type	○	proposition	○	theorem proving
Mallya et al.* [66, 68]	●	●	●		abstract data type	○	proposition	○	
Gerard et al. [59]	●	●	●	●	object	○	proposition	○	model checking
Telang et al. [90]	●	●	●	●	module	○	proposition	○	model checking
Kafali et al. [61]	●	●	●	●	module	○	proposition	○	model checking

communicated. The third claim is called *practical claim* and means that the debtor agent is justified by doing the communication. An example of objective semantics is as follows:

Example 2.3.7. *An informative message* $inform(Mer, Cus, sendReceipt)$ *can be defined as a commitment* $C(Mer, Cus, sendReceipt)$ *with its sender agent* Mer *as debtor, its receiver agent* Cus *as creditor and its content as commitment consequence asserting the truth of the proposition* $sendReceipt$.

To formalize the defined meaning claims, the author extended CTL with commitment modality, belief modality, and intention modality. Three accessibility relations are presented to define the semantics of these modalities. For instance, the accessibility relation of social commitments is defined as: $\mathcal{C}$: $\mathcal{A}\times\mathcal{A}\times\mathcal{A}\times S \rightarrow 2^{\Pi}$ where $\mathcal{A}$ is a set of agents and Π is a set of paths. It specifically computes the set of accessible paths along which the commitments made by i at a state $s \in S$ towards j in a certain context Con hold. Given that, the semantics of the commitment $C(i, j, Con, p)$ is satisfied in a model M at s iff the consequence p is true along every accessible path π defined by $\mathcal{C}(i, j, Con, s)$ and started at the commitment state s in the context Con:

$$M, s \models C(i, j, Con, p) \textit{ iff } \forall \pi : \pi \in \mathcal{C}(i, j, Con, s) \rightarrow M, \pi(s) \models p$$

In order to define the semantics of conditional commitments, the strict implication operator $\rightsquigarrow$ is proposed and used to extend CTL as well. The semantic rule of the strict operator is given as follows:

$$M, s \models p \rightsquigarrow q \; iff \; M, s \models p \; and \; \forall s' : M, s' \models p \rightarrow (\forall s'' : s' \approx s'' \rightarrow M, s'' \models q)$$

Semantically, the state formula $p \rightsquigarrow q$ is satisfied in a model M at s iff p holds in the current state s and at all states $s' \in S$ in M, if p holds, q will then hold in all states s'' which are similar to s' (i.e., $s' \approx s''$). An example of subjective and practical semantics is as follows:

Example 2.3.8. *The subjective and practical semantics of* $inform(Mer, Cus, sendReceipt)$ *are defined as* $C(Mer, Cus, NB, MerBsendReceipt)$ *and* $C(Mer, NB, inform(Mer, Cus, sendReceipt) \rightsquigarrow sendReceipt)$ *where* $MerBsendReceipt$ *means* Mer *beliefs* $sendReceipt$ *in the NetBill protocol context. According to the semantics of the strict operator, the proposition* $sendReceipt$ *will hold only if the satisfaction of* $inform(Mer, Cus, sendReceipt)$ *is true.*

Singh's semantics satisfies formal, declarative, and meaningful criteria. The author considered a 'conversation for action protocol' as a motivating example to illustrate the importance of social semantics without discussing any experimental results.

Limitations:

1. While the commitment semantics criterion is met, the proposed semantics for unconditional and conditional commitments are not intuitive and computationally non-grounded because the intuitive relation between a social commitment and accessible paths as well as how to compute accessible paths within a computational model are not clear.
2. Formal semantics of unconditional and conditional commitment actions are not considered.
3. It is very difficult to verify the correctness of the introduced semantics as it is given in terms of mental states as shown in [99].

The authors presented in [51] a framework that comprises of three parts. In the first part, they introduced a CTLC logic, an extension of CTL with modality to represent and reason about unconditional commitments. Then, they defined a social accessibility relation R_{sc} to define the semantics of commitments. It is the first framework in commitment-based approaches, which uses the formalism of interpreted systems [57] to: 1) model the internal components of agents such as local states, local actions, and local policy as well as model global behavior of the system from local components; and 2) generate the temporal model of CTLC. Therefore, the standard CTL model $M = (S, T, V, I)$ is extended to $M' =$

(S, T, R_{sc}, V, I) where $R_{sc} : S \times \mathcal{A} \times \mathcal{A} \rightarrow 2^S$ is the social accessibility relation. This relation is defined as follows:

$$s' \in R_{sc}(s, i, j) \textit{ iff } \exists \overline{s} \in S : 1)\ l_i(s) = l_i(\overline{s}) \textit{ and } 2)\ l_j(\overline{s}) = l_j(s')$$

Here, $l_i(s)$ denotes the local state of agent i in the global state s. This definition is intuitive and shows how to compute accessible states, which are missing in [79]. s' is intuitively accessible from s (i.e., $s' \in R_{sc}(s, i, j)$) iff there is a state $\overline{s}$ such that it is indistinguishable for the debtor i between being in s and $\overline{s}$; but, for the creditor j there is no difference between being in the intermediate state $\overline{s}$ and the accessible state s'. In the second part, the semantics of commitments is defined as follows:

$$M', s \models C(i, j, \varphi) \textit{ iff } \forall s' \in S,\ \textit{if } s' \in R_{sc}(s, i, j),\ \textit{then } s' \models \varphi$$

The commitment formula is satisfied at s in M' if the consequence φ is true in every accessible state from the current state computed using $R_{sc}(s, i, j)$. In the last part, the problem of model checking CTLC is formally transformed into the problems of model checking CTLK (an extension of CTL with knowledge modality [75]) and ARCTL (an extension of CTL with action formulae [73]). This transformation allows the authors to respectively use the MCMAS tool and extended NuSMV tool [63]. The authors also verified the correctness of the Contract Net protocol against some desirable properties, such as *reachability, liveness* and *safety*, for example:

Example 2.3.9. *Reachability property: given a particular state, is there a valid computation sequence to reach that state from an initial state. The following formula is used to check that the 'Payment' state is reachable from the 'Goods' state in the NetBill protocol:* $E(\neg Goods\ U\ (Goods \wedge C(Cus, Mer, sendPayment)))$.

The authors considered only 4 agents. It is clear that the proposed semantics meets our criteria, except commitment action-oriented criteria.

Limitations:

1. This framework considered only unconditional commitments.
2. Unconditional and conditional commitment actions are not considered.

The authors improved their CTLC logic introduced in [51] with two temporal modalities to represent and reason about fulfillment (Fu) and violation (Vi) of commitments [53]. Given that, they proceeded to develop a symbolic algorithm to perform the model checking problem of the presented logic; instead of transforming the problem of CTLC model checking

into another existing problem. Such an algorithm is implemented on top of the MCMAS model checker [64]. The authors considered the NetBill protocol as a case study with 20 agents. An example of a formula including the fulfillment modality is as follows:

Example 2.3.10. $EF\ Fu(C(Cus, Mer, sendPayment))$*, meaning that there is a path in its future* Cus *fulfills its commitment by sending the promised payment to* Mer.

After one year, Bentahar et al. [15] and El Menshawy et al. [52] redefined: 1) the social accessibility relation introduced in [53] in order to account for the intuition that social commitments are conveyed through communication between interacting agents; and 2) the semantics of commitment and fulfillment modalities. In terms of time and space complexity aspects, the authors removed the violation modality introduced in [53]. The authors particularly used commitment and fulfillment temporal modalities along with CTL modalities to express the violation of commitments in the form of temporal properties. An example of the violation property is as follows:

Example 2.3.11. *The violation property is valid when there is a computation path so that in its future a commitment to deliver the requested goods is activated but from the moment where the commitment is active, its fulfillment never happens along all possible computations:* $EF(C(Mer, Cus, deliverGoods) \wedge AG(\neg Fu(C(Mer, Cus, deliverGoods))))$.

The refined logic is so-called CTLC$^+$. For the verification process, Bentahar et al. [15] adopted the direct verification technique by developing symbolic algorithms needed for the added modalities. On the other hand, El Menshawy et al. [52] formally reduced the problem of model checking CTLC$^+$ into the problem of model checking ARCTL and the problem of model checking GCTL* [18] (a generalized version of CTL* with action formulae). This reduction process allows the authors to use the extended NuSMV symbolic model checker and the CWB-NC automata-based model checker[1]. The authors in [52] considered the NetBill protocol and the Contract Net protocol as two case studies with varying size of agents ranging from 2 to 6 agents. The authors also expressed in CTLC$^+$ a set of desirable properties such as *safety* and *reachability* to automatically check the correctness of their protocols, for example:

Example 2.3.12. *The safety property states that "something bad never happens". The bad situation happens when* Mer *fulfills its commitment by delivering the requested goods, but* Cus *never commits to send the payment:*
$AG\neg(Fu(C(Mer, Cus, deliverGoods) \wedge AG\neg\ C(Cus, Mer, sendPayment)))$.

[1] `http://www.cs.sunysb.edu/cwb/`

To come into the evaluation of [15, 52], the semantics of these proposals meets our criteria, except commitment action-oriented criteria.

Limitations:

1. This framework considered only unconditional commitments.
2. There is no semantics for other unconditional commitment actions.
3. The semantics of the fulfillment modality implies that the unconditional commitment is active at the moment of its fulfillment: $Fu(C(i,j,\varphi)) \rightarrow C(i,j,\varphi)$. This paradox is not commonly accepted in the literature (see for example [81, 97, 106]).

The authors in [3] combined CTLK (an extension of CTL with knowledge modality [75]) and CTLC$^+$ introduced in [15, 52] in order to study the logical relationship between agents knowledge and commitments. By expressing and figuring out a set of reasoning postulates, which include knowledge and commitments together, they identified a set of paradoxes. Some of these paradoxes are solved by relaxing the conditions of the accessibility relation introduced in [15, 52] and other paradoxes are addressed by adding other conditions. All these solutions conducted the authors to produce another logic called CTLKC$^+$. They also adopted a reduction technique [4], which formally transforms the problem of model checking CTLKC$^+$ into the problem of model checking an existing logic of action called ARCTL in order to get benefit from the extended version of NuSMV dedicated to ARCTL. On the same direction of research, the logical relationship between probability and unconditional commitments and their fulfillments has been studied in [88, 87] in a logic called probabilistic computation tree logic of commitments (PCTLC). In fact, PCTLC is a combination of the probabilistic extension of CTL called PCTL and CTLC$^+$ introduced in [15, 52]. The problem of model checking PCTLC is transformed into the problem of model checking PCTL to take benefit of existing model checker PRISM. The authors considered the the oblivious transfer protocol (OTP), taken from the cryptography domain, as a case study. In their simulation results, they considered a system of 24 agents.

Limitation: since the proposals of Faisal et al. and Sultan et al. are based on the CTLC$^+$ logic introduced in [15, 52], these proposals share the same evaluation and limitations with regard to our criteria.

The following table (Table 3) summarizes our results of evaluating the extended CTL-based approaches. Except [79] in the table, other approaches considered only unconditional commitments and their fulfillment actions.

Table 3: Summary of extended CTL-based approaches

	Agent Communication Semantics				Commitment		Action		Verific. Method
	For.	Dec.	Mea.	Ver.	Mod.	Sem.	Mod.	Sem.	
Singh [79]	●	●	●	●	temporal modality	●	○	○	model checking-like
El Menshawy et al. [51]	●	●	●	●	temporal modality	●	○	○	model checking (formal translation to MCMAS and NuSMV)
El Menshawy et al. [53]	●	●	●	●	temporal modality	●	temporal modalities for Fu and Vi	●	dedicated a new model checking algorithm
Bentahar et al. [15]	●	●	●	●	temporal modality	●	temporal modality for Fu	●	dedicated a new model checking algorithm
El Menshawy et al. [52]	●	●	●	●	temporal modality	●	temporal modality for Fu	●	model checking (formal translation to extended NuSMV and CWB-NC)
Faisal et al. [4]	●	●	●	●	temporal modality	●	temporal modality for Fu	●	model checking (formal translation to extended NuSMV)
Sultan et al. [87]	●	●	●	●	temporal modality	●	temporal modality for Fu	●	model checking (formal translation to PRISM)

2.3.3 Summary of limitations

We conclude this chapter by summarizing the current limitations as follows:

1. The current semantics of fulfilling unconditional commitments leads to counterintuitive situations (or paradox).
2. There is no suitable semantics for conditional commitments in the current CTL-based approaches.
3. There is no formal semantics for conditional commitment actions in the current CTL-based approaches.
4. With the strict implication operator used to model conditional commitments [79], we cannot distinguish between different types of conditional commitments.
5. The current implementation clearly demonstrates very limited scalability in terms of considered agents along with high memory consumption.

In Chapter 3, we propose a new temporal logic system to address these limitations by extending the CTL modalities with two types of conditional commitment modalities and conditional commitment action modalities. Moreover, we develop a new symbolic algorithms for conditional commitments and all associated actions in Chapter 4. We also prove the memory usage and running time of our algorithm are polynomial in Chapter 5 and its scalability is high in terms of number of considered agents in Chapter 6.

Chapter 3

Temporal Logic of Conditional Commitments and their Actions

This chapter[1] covers:

- An overview about the proposed approach, a theory of conditional commitments, motivation examples, and commitment life cycle.
- The formalism of interpreted systems [57] and our extended version of this formalism [39, 52].
- The syntax and semantics of the developed CTL$^{cc,\alpha}$ logic, which addresses Questions 1.3.1, 1.3.2, 1.3.3, and 1.3.4 mentioned in Chapter 1.
- A set of properties of CTL$^{cc,\alpha}$, a set of reasoning rules, and a set of action postulates, which address Question 1.3.5 mentioned in Chapter 1.

3.1 An overview of the proposed approach

Despite the popularity of conditional commitments in multi-agent communication research, they are yet to be seen to work well in real and concrete applications. We believe that the fundamental road to success for conditional commitments is in an operational approach, which ought to be rich and expressive enough to accommodate business scenarios of feasible utility and address the limitations identified in Chapter 2 in a reasonable and rigourous way along with its ability to supply model checking at design time [32]. Figure 3 illustrates the main parts of the proposed approach along with our contributions in the corresponding

[1] The results in this chapter are collected from our publications in [39, 41, 44].

chapters. It specifically crosses over five different, but fully integrated and interoperable, parts.

1. In the logical language part (which is the core of this chapter), we develop a new branching-time temporal logic called CTL$^{cc,\alpha}$. This logic is an extension of the CTL logic [32] with new modalities for two types of conditional commitments and common conditional commitment actions. What makes the commitment language CTL$^{cc,\alpha}$ special is:

   – Effectively modeling agent interactions (commitments) as temporal modalities;
   – Effectively modeling agent conditional commitment actions as temporal modalities;
   – Accurately expressing state-based properties, including temporal commitment properties; and
   – Accurately expressing conditional commitment action-based properties.

   By modeling agent interactions as temporal modalities, we: 1) have a concrete and intuitive semantics that can be model checked at design time for conditional commitments and conditional commitment actions; and 2) address the limitations of modeling unconditional commitments as fluents [23] and predicates [106] (see Chapter 2). Expressing only state-based properties as in all CTL-based approaches discussed in Chapter 2 limits to some extent the expressive power of the logical language, and thus restricts the specifications to only propositional messages without being able to consider action messages that agents can perform. Moreover, the current main approaches (mental and social) have concentrated on the semantic part of agent communication by capturing pre- and post-conditions needed to exchange messages among agents without capturing in a suitable way the dynamic interactions. The following example clarifies this limitation:

   Example 3.1.1. *From the recent semantics of FIPA-ACL messages called FIPA-SL*[2]*, the mental semantics of the performative* $inform$ *is defined as follows:*

   $\langle i, inform(j, \varphi)\rangle$

   feasibility precondition: $B_i\varphi \wedge \neg B_i(Bf_j\varphi \vee Uf_j\varphi)$

   rational effect: $B_j\varphi$

[2]FIPA SL content language specification: http://www.fipa.org/specs/fipa00008/index.html

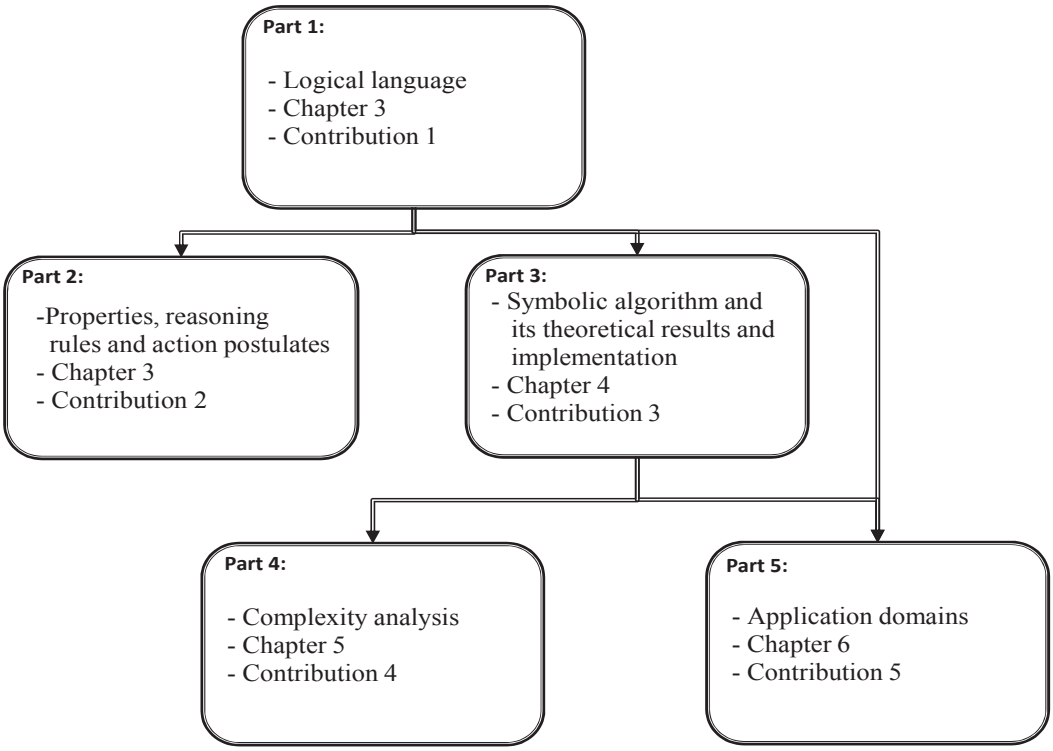

Figure 3: The main parts of the proposed approach

This semantics is formally defined by a quantified multi-modal logic with several referential quantifiers (e.g., any and all) and action operators (e.g., feasible and done). Precondition: agent i informs agent j that φ, meaning i believes φ and doesn't believe that: 1) j believes φ or its negation; or 2) j is uncertain about the truth or falsity of φ. Postcondition (or the rational effect): the receiver agent j will believe φ.

From the designer perspective, a transition beginning at the precondition state and ending at the postcondition state exists, but it is not well-defined, because the system's states change without reflecting explicitly the source reasons that cause the transitions to be taken. Thus, the transitions should be labeled with guarded conditions (actions) needed to fire these transitions, provided the satisfaction of preconditions. In fact, the natural way of modeling interaction among intelligent agents is through performing actions as outlined in the philosophical *speech act theory* [7], the cornerstone of agent communication theory: "Almost any speech act is really the performance of several acts at once". The core of this theory is the *illocutionary act*, which captures the basic principle indicating that "by saying something, we do

something".

2. In the second part, we present a set of valid properties, a set of reasoning rules, and a set of action postulates in order to explore the capabilities of CTL$^{cc,\alpha}$ in the current chapter. Furthermore, we propose a new life cycle of conditional commitments in the same part.

3. In the third part (symbolic algorithm and its theoretical results and implementation), we start by developing a new symbolic algorithm to tackle the problem of model checking CTL$^{cc,\alpha}$. Instead of developing our algorithm from scratch, we extend the standard CTL model checking algorithm introduced in [32] with symbolic algorithms needed for our new modalities (see Chapter 4). In fact, symbolic techniques alleviate the state explosion problem, but cannot eliminate it totally as the space still increases when the model is getting larger [32]. In this part, we also investigate important theoretical results of our algorithm: soundness and termination. In this part, we completely implement our algorithm and then assemble it on top of the symbolic model checker MCMAS [64] (see Chapter 4). MCMAS is developed to automatically and directly test MASs specifications. Such an implementation is not an obvious task nor trivial as we need first to encode Boolean formulae representing all components of the model as well as the states and sets of states computed by the developed algorithm in OBDDs. We then add our symbolic algorithms along with other modifications needed for the implementation interest. The resulting symbolic model checker is so-called MCMAS+. We also extend MCMAS's input modeling and encoding language called ISPL with shared and unshared variables needed for agent interactions and with the syntactic grammar of new modalities in order to produce a new one called ISPL+. In addition to MCMAS+ inherits the powerful capabilities of MCMAS, we develop and implement Java modules, which extend the MCMAS graphical user interface to: 1) consider our new modalities; and 2) display models. The latter extension in fact improves the usability of the graphical user interface.

4. In the complexity analysis part in Figure 3, we analyze the time and space complexity of the developed algorithm in the third part. A positive result of our model checking algorithm is that its analyzed time complexity is P-complete for explicit models and its analyzed space complexity is PSPACE-complete for concurrent programs (see Chapter 5). This is because model checking CTL$^{cc,\alpha}$ has the same time and space computational complexity of model checking CTL although CTL$^{cc,\alpha}$ extends CTL. We utilize a reduction technique to compute the upper and lower bounds

of the problem of model checking $CTL^{cc,\alpha}$. The motivation behind considering the complexity of our model checking algorithm for concurrent programs that provide compact representations is that explicit representations where states and transitions are listed explicitly (as Kripke-like structures) are not supported in practice by actual model checking tools. In particular, practical model checkers such as MCMAS (for CTL), NuSMV (for CTL), SPIN (for LTL), and CWB-NC (for $GCTL^*$ and CTL^*) have the flavour of compact modeling languages that disagree upon details and provide the tool with a relatively high-level method of defining concurrent programs. The explicit models are obtained as the product of the components of concurrent programs. Thus, the size of explicit models is exponential in the size of components, since the evolution of the system results from joint actions of the components, as we shown in [52].

5. In the application part, we illustrate the feasibility, efficiency and scalability of our approach through a set of reasonable and business-oriented case studies (see Chapter 6). Such case studies are taken from the business protocol domain, health care domain, and web service composition domain. In each application, we will help designers to model, express properties in $CTL^{cc,\alpha}$, encode, and verify the application correctness. We also report our experimental results, evaluate the effectiveness of the developed algorithm using different interaction techniques, and compare the experimental results with existing approaches (if any).

3.2 Conditional commitments, motivation examples and commitment life cycle

3.2.1 The theory of social commitments

Commitments socially represent the engagement of one software intelligent agent or party with another party. The party who "owes" the commitment is termed the debtor and the other party is termed the creditor [83]. That is, a commitment represents a peer-to-peer interaction pattern, which is started by her debtor and directed towards her creditor. Every party looks for some profits from the commitment and is ready to receive some obligations in order to incur them. Commitments provide a method to precisely represent, reason, and model software agents inter-interactions in the form of obligatory contracts [35, 38]. Social and objective commitments have been the subject of considerable research and witnessed

a huge growth in the literature of agent communication over the past two decades, as on the one hand, they preserve agents flexibility and autonomy [9, 15, 49] and on the other hand provide an effective way to qualify the degree of autonomy and interdependency of software agents [52, 83, 103]. They also provide a good basis for verifying the compliance of agents' behaviors with specifications [15, 53, 52]. There are two sorts of social commitments: *unconditional* and *conditional* (see for example [9, 36, 52, 49]). Unconditional commitment has the shape $C(Dt, Ct, Csq)$ where Dt is her debtor, Ct her creditor, and Csq the consequence the debtor will bring about. Conditional commitment has the form $CC(Dt, Ct, Ant, Csq)$ wherein the debtor can only devote to the creditor to bring about some consequences (Csq) as long as some antecedents Ant are met. In the theory of social commitments, agents' behaviors can be understood in terms of how they affect the social state of their active commitments. Such dynamic behaviors are captured by a set of legal actions called commitment actions. There are duplex actions (fulfill, cancel, and release), which have two parties and triplex actions (assign and delegate), which include three parties. These actions enable us to define the life cycle of conditional commitment: a set of states connected by directed transitions, which are labeled with duplex/triplex actions.

While current approaches do not distinguish between different but related types of conditional commitments, in the logical language part of our approach, we distinguish strong conditional commitments from classical or weak ones. Concretely, weak commitments are those that can be activated even if the antecedent will never be satisfied, while strong commitments are solely activated when there is a possibility to satisfy their antecedents. Keeping in mind that as the focus is on model checking, the model is entirely known because the real system is abstracted into a fully analyzable finite state model[3]. To demystify the motivation behind distinguishing between weak and strong conditional commitments and giving an importance for introducing strong conditional commitments, let us examine the following examples.

3.2.2 Motivation examples for weak and strong commitments

We start with a key business scenarios taken from the NetBill protocol.

Example 3.2.1. *Consider the NetBill protocol [84] as modeled using event calculus in [104, 106]. Social commitments conventionally allow us to flexibly specify the protocol. For instance, they enable us to begin the interaction by: 1) a merchant commits to present*

[3]The fact that the model has finite states and infinite transitions is a common assumption in the literature of model checking approaches (see for example [32]).

an offer without receiving a request from a customer (as it happens for advertising); 2) a merchant can commit to deliver some goods for trial without asking a customer for accepting the price; or 3) a customer commits to accept the price quote before the merchant proposes one, mimicking the customer's trust on the fact that the merchant will make an offer.

From the example, it is not suitable to presume that the interaction can be relied on the trust that the agents have toward the others. Agents are indeed heterogeneous and therefore there are no guarantees on how they are implemented (e.g., how to distinguish malicious agents?). Suppose the customer has some reasons for trusting the merchant, what takes place when the merchant is willing but is practically unable to present the offer? Therefore, the distinction between strong and weak conditional commitments would help us address the shortcomings regarding the temporal ordering between the acquisition of antecedents and consequences of commitments (cf. Definition 3.4.3 for the defined formal semantics). More specifically, if we model the conditional commitment about accepting the price quote by the customer as a strong one, it will not be active (i.e., it will not exist) until there is at least one possibility in the agent's model to receive the offer by the merchant. However, it is not sufficient to modify the commitment antecedent by modeling that the commitment is in place when there is a request to do so and there is one available (as in Singh's modeling using the strict operator [79]), because we will: 1) lose the flexibility of social commitment-based approaches according to which a commitment can take place even if there is no chance to satisfy the antecedent (which is commonly agreed on the literature and reflected in the previous operational semantics of conditional commitments, see for example [104, 106, 97]); and 2) not be able to take advantage of opportunities or to handle exceptions that could appear during interactions. For these reasons, we introduce weak commitments on top of strong commitments, which for instance can be exploited to model the conditional commitment about presenting the offer (i.e., a commitment will be in place) even if we know that the antecedent will never be satisfied (i.e., there is no way to send a request by the customer).

Now, we consider the following examples taken from the health care setting. Some of these examples are discussed in [81, 70].

Example 3.2.2. *An insurance company strongly commits to reimburse a covered patient for a health procedure provided the patient obtains an approval from the company prior to the health procedure. Otherwise, the patient would delay going in for the procedure until she gets an approval. The commitment is strong because there is always a way to obtain*

the approval.

Example 3.2.3. *A physician devotes to a patient that if the patient has any sign of heart trouble after signing up with her, the patient will be immediately referred to a laboratory for tests, the results of which will be evaluated by a specialist. The commitment is strong as there is a chance that the antecedent will be met.*

Example 3.2.4. *A pharmacy strongly commits to provide medicine only if the patient pays and shows a prescription for it, an antecedent which is always possible.*

Example 3.2.5. *A surgeon cannot strongly commit to replace an organ unless there is assuring possibility to find a good one (thinking about getting liver transplants).*

As a result, strong commitments are very significant and capture real and cooperative situations that weak commitments cannot suitably model. Also, they often give more confidence in terms of their fulfillments than the weak ones. For instance, when the good organ exists, then the fulfillment degree of the surgeon's commitment to replace the defective one is very high.

3.2.3 Conditional commitment life cycle

In our state machine (also called *conditional commitment life cycle*), a conditional commitment (either strong or weak) can be held in one social state and remains in this state until a duplex/triplex action is directly applied to it. Figure 4 depicts a conditional commitment life cycle in a directed graph where solid arrows represent the transitions from a commitment state to another and dashed arrows represent the creation of a new commitment induced by delegate and assign actions on an active commitment. We initially presume any potential conditional commitment is in the *Not Active* state. When the creation action is performed, the commitment state turns to the *Active* state. The active commitment can be in one of the following states.

1. When commitment is fulfilled by her debtor, the commitment becomes no longer active. The commitment state is then changed from the *Active* state into the *Fulfilled* state.
2. When commitment is revoked by her debtor, the commitment becomes no longer active and her *Active* state will be moved to the *Canceled* state.
3. When commitment is released by her creditor to free her debtor, the commitment becomes no longer active and her *Active* state will be moved to the *Released* state.

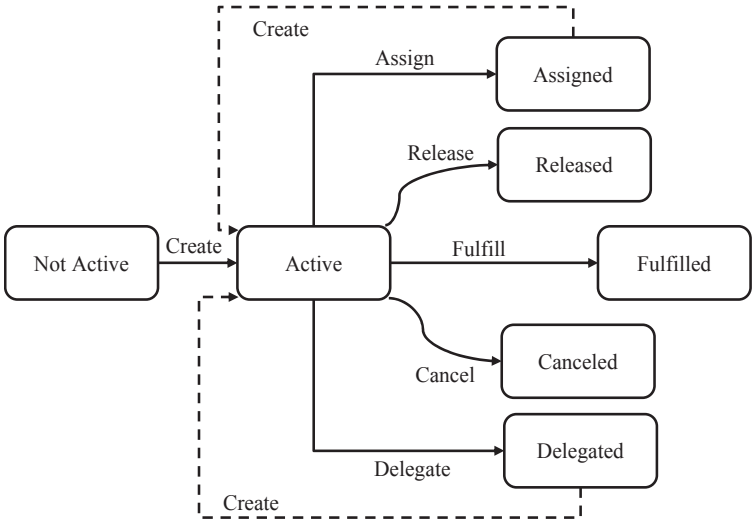

Figure 4: Conditional commitment life cycle

4. When commitment is delegated by her debtor to shift her role to another software agent called delegatee, then the delegatee agent creates a new commitment on behalf of the delegator. The *Active* state is moved to the *Delegated* state.
5. When commitment is assigned by her creditor to transfer her commitment to another software agent called assignee, then the assignee becomes the creditor of the new commitment. The *Active* state is moved to the *Assigned* state.

Several approaches have discussed the social commitment life cycle [23, 49, 82]. However, these approaches require first to detach conditional commitments as long as their antecedents are satisfied into unconditional ones and then apply commitment actions to the detached ones (i.e., unconditional commitments). For instance, the recent life cycle introduced by Chesani et al. [23] (see Figure 2, page 97 in this reference) doesn't have the possibility to apply commitment actions (e.g., discharge or fulfill action) to conditional commitments directly and this is why the formal semantics of conditional commitment actions are yet to be investigated. Therefore, our life cycle is the first proposal that formally applies commitment actions directly to conditional commitments. This life cycle is implemented by local and social transition functions in our case studies in which these commitment actions are defined in the model of interacting agents as local communicative actions (see Section 3.3 for the formalization of these functions). Concretely, such an implementation considers the changes that a conditional commitment might go through from

activation to termination. Moreover, our life cycle is flexible as it can support any interaction pattern in real life scenarios subject to a mapping relation, such as the "count as" relation in Searel's theory [78] and the "means" construct in [26] and [11]. The mapping relation in principle captures the social meaning of performing a specific action within a certain context. For example, we can directly map protocol messages such as 'deliver goods' message into a 'fulfill' action in a direct and natural way.

3.3 Extended version of interpreted systems

The formalism of interpreted systems [57] provides a very popular mainstream framework for modeling and reasoning about MASs. This framework locally models autonomous and heterogeneous agents who interoperate within a global system via sending and receiving messages [15, 53, 52]. The interactions among agents are implemented by the use of message-passing systems. In [39, 52], we extended this formalism with sets of shared and unshared variables to account for interactions that occur during the execution of MASs. These variables play an essential role in the definition of the social accessibility relation. In the extended version of interpreted systems, a MAS is a composed set of software intelligent agents $\mathcal{A} = \{1, \ldots, n\}$ plus the environment agent e.

– Each software agent $i \in \mathcal{A}$ is described by:

1. A set L_i of instantaneous and finite local states. Each private local state l_i represents the whole information about the system that the agent has at a given moment.
2. A set Act_i of finite local actions available to the agent, including the conditional commitment actions $Act_i^c = \{Fulfill, Cancel, Release, Delegate, Assign\}$ and $null$ action, which refers to the fact of doing nothing. Differently from L_i, Act_i is public and accounts for the temporal evolution of the system.
3. A set Var_i of at most $n-1$ local variables to model communication channels with all other agents. Through such channels values are sent and received as in distributed systems. Intuitively, $|Var_i| \leq n-1$ as i might not have communication channels with particular agents.
4. A local protocol function $\mathcal{P}_i : L_i \rightarrow 2^{Act_i}$, which represents the decision making procedure of i and produces the set of enabled actions that might be performed by i in a given local state. When more than one action is enabled, it is assumed that an agent selects *non-deterministically* which action to perform.

5. A set $\iota_i \subseteq L_i$ of initial states.
6. A local transition function $\tau_i : L_i \times Act_i \rightarrow L_i$, which defines the evolution from a local state to another local state given a local action.

– An environment agent e: agents interact with each other and with an "environment" e. This can be seen as a special agent, which can capture any information that may not pertain to a specific agent. It is modeled by $L_e, Var_e, Act_e, \mathcal{P}_e, \iota_e$ and τ_e with the above meanings. The local states in L_e are public, i.e., all the remaining agents can access.

– Having described the formalism of $n + 1$ agents, the MAS system can be described as follows:

1. The notion of social state (termed global state in [57]) represents the instantaneous configuration (or screenshot) of all agents in the MAS system at a certain moment. A social state $s \in S$ is a tuple $s = (l_1, \ldots, l_n, l_e)$ where each element $l_i \in L_i$ represents the i's local state along with the environment state l_e.
2. The set of all social states $S \subseteq L_1 \times \ldots \times L_n \times L_e$ is a subset of the Cartesian product of all local states of all agents and the environment agent.
3. A social transition function is defined as $\tau : S \times ACT \rightarrow S$, where $ACT = Act_1 \times Act_2 \times \cdots \times Act_n \times Act_e$ and each component $a \in ACT$ is a joint or shared action, which is a tuple of actions (one for each agent).
4. Let $l_i(s)$ denotes the local state of the software agent i in the social state s and the value of a variable x in the set Var_i at $l_i(s)$ is denoted by $l_i^x(s)$. When $l_i(s) = l_i(s')$, then for all $x \in Var_i$ we have $l_i^x(s) = l_i^x(s')$.
5. To allow agent i to communicate with agent j, they should share a channel, which is represented by a shared variable between them. Formally, a communication channel between i and j coexists as long as $|Var_i \cap Var_j| = 1$. For the variable $x \in Var_i \cap Var_j$, $l_i^x(s) = l_j^x(s')$ means that the values of x in $l_i(s)$ for i and in $l_j(s')$ for j are the same. Intuitively, the existence of a communication channel between i in s and j in s' means the value of the variable x has been sent by one of them towards the other, therefore, i and j will have the same value for this variable as a consequence of the communication between them.
6. The valuation function $\mathcal{V} : \mathcal{AP} \rightarrow 2^S$ defines what atomic propositions are true from the set $\mathcal{AP}$ at system states.

In summary, the extended version of the formalism of interpreted systems is denoted by the tuple $IS^+ = \big(\{L_i, Var_i, Act_i, \mathcal{P}_i, \tau_i, \iota_i\}_{i \in \mathcal{A}}, \{L_e, Var_e, Act_e, \mathcal{P}_e, \tau_e, \iota_e\}, \mathcal{V}\big)$. It easy to see

that such a formalism provides a modular representation in the sense that components can be replaced, removed or added with little modifications to the whole representation [46].

3.4 The CTL$^{cc,\alpha}$ logic

The semantics of CTL$^{cc,\alpha}$ formulae is interpreted using a model generated from the extended version of interpreted systems discussed above.

Definition 3.4.1 (CTL$^{cc,\alpha}$ models)**.** *A temporal model* $M = \big(S, I, T, \{\sim_{i\to j} \mid (i,j) \in \mathcal{A}^2\}, \{R_{c_i} \mid i \in \mathcal{A}\}, \mathcal{V}\big)$ *is generated from* $IS^+ = \big(\{L_i, Var_i, Act_i, \mathcal{P}_i, \tau_i, \iota_i\}_{i\in\mathcal{A}}, \{L_e, Act_e, \mathcal{P}_e, \tau_e, \iota_e\}, \mathcal{V}\big)$ *by synchronising joint actions of* $n+1$ *composed agent models as follows:*

- S *is a set of social states for the system.*
- $I \subseteq \iota_1 \times \ldots \times \iota_n \times \iota_e$ *is a set of initial states for the system such that* $I \subseteq S$.
- $T \subseteq S \times S$ *is a total temporal relation (i.e., each state has at least one successor) defined by* $(s, s') \in T$ *iff there exists a joint action* $(a_1, \ldots, a_n, a_e) \in ACT$ *such that* $\tau(s, a_1, \ldots, a_n, a_e) = s'$.
- *For each pair of software agents* $(i,j) \in \mathcal{A}^2$, $\sim_{i\to j} \subseteq S \times S$ *is a social accessibility relation defined by* $s \sim_{i\to j} s'$ *iff the following conditions hold:*
  1. $l_i(s) = l_i(s')$.
  2. $(s, s') \in T$.
  3. $Var_i \cap Var_j \neq \emptyset$ *and* $\forall x \in Var_i \cap Var_j$ *we have* $l_i^x(s) = l_j^x(s')$.
  4. $\forall y \in Var_j - Var_i$ *we have* $l_j^y(s) = l_j^y(s')$.
- *For each software agent* $i \in \mathcal{A}$, $R_{c_i} \subseteq L_i \times Act_i^c \times L_i$ *is a local labeled transition relation defined by* $(l_i(s), a_i, l_i(s')) \in R_{c_i}$ *if* $\tau_i(l_i(s), a_i) = l_i(s')$.
- $\mathcal{V} : \mathcal{AP} \to 2^S$ *is a valuation function defined as in* IS^+.

Similar to standard CTL model, the model M conceptualizes time as a tree-like structure whose nodes correspond to the states of the system being considered. The paths of the tree represent all choices in the future that agents have when they participate in conversations and protocols, while the past is linear. Concretely, the underlying time domain in M is discrete, i.e., the present moment refers to the current state, the next moment corresponds

to the immediate successor state in a given path and a transition represents a social interaction between agents and corresponds to the advance of a single time-unit. This means that there is no notion of real time, while reasoning about discrete time is possible through state variables that keep track of time and count transition steps. Notice that $\sim_{i\to j}$ captures the intuition that for a conditional commitment to take place, a communication channel should exist between i and j through the shared variable, and the accessible state s' (where φ and ψ hold) is reachable in one step using T[4] and is indistinguishable from the current state s for i as i is the agent who is committing (which reflects the persistence of i towards its commitment); however, for j who is receiving the commitment, the two states are different as new information is obtained from i through the communication channel and this is why in the accessible state, j has the same value as i has for the shared variable (i.e., the content of the communication channel). Furthermore, the accessible state is not completely different from the current state for j as some information is still the same, and this is why for the unshared variables, the current and accessible states for j are indistinguishable. Moreover, R_{c_i} is in fact the projection of τ by τ_i labeled by $a_i \in Act_i^c$. It enables us to locally label transitions with commitment actions applied to active commitments and to formally define the intuitive and reasonable semantics of commitment action modalities.

Figure 5 illustrates an example of establishing a communication channel among two

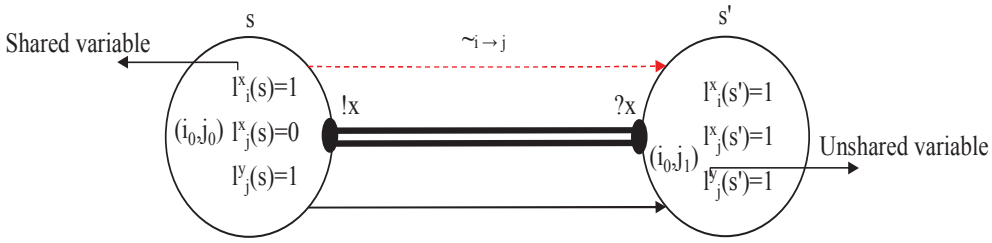

Figure 5: An example of accessibility relation $\sim_{i\to j}$ where ! and ? refer to sending and receiving value.

social states: $s = (i_0, j_0)$ and $s' = (i_0, j_1)$. In this example, two agents are communicating and the shared and unshared variables for these agents are as follows. Agent i: $Var_i = \{x\}$. Agent j: $Var_j = \{x, y\}$. In the figure, x is the shared variable (i.e., it represents the communication channel between i and j), and y is a j's variable unshared with i. The value of the variable x for j in the state s' is changed to be equal to the value of this variable

[4]This condition is not only technical, but also practical. Its practical aspect is that the commitment is one step-moment consuming. This moment is needed for sending the message to the addressee.

for agent i, which illustrates the sequence: sending value, passing through the channel, and receiving value. However, the value of the unshared variable y is unchanged. Notably, the communication between two social states is not constrained to two agents, as the communication can be established between n agents at two social states using $n-1$ shared variables representing communication channels for each agent. For example, suppose we have agents $\{i, j, k\}$ communicating together to form a MAS system, then each social state will contain 3 agents. In this system, i can communicate with j using shared variable x and i can communicate with k using shared variable y. Similarly, j can communicate with k and i using shared variables z and x. The communication channels of k can be represented by y and z. Furthermore, the proposed approach supports synchronous communication because our extended version of the interpreted system formalism is like the original version, which is composed by the Cartesian product of the set of $n+1$ agents synchronized by joint actions. However, supporting synchronous communication is independent from the shared variables, which merely motivate the existence of channels for communication. This explains why the communication channels don't change over time as they are independent from the communication content itself.

3.4.1 Comparing models of CTL$^{cc,\alpha}$ and CTLC^{+}

At the first glance, the models of CTL$^{cc,\alpha}$ and CTLC^{+} introduced in [15] seem to be similar. In this section, we will show that they have different semantics. Particularly, these two models are not equivalent with respect to the definition of the accessibility relation. On the one hand, the accessibility relation in CTLC^{+} is defined as follows: $s \sim'_{i\to j} s'$ iff 1) $l_i(s) = l_i(s')$; 2) there exists at most one shared variable x, denoted by $\exists! x \in Var_i \cap Var_j$, such that $l_i^x(s) = l_j^x(s')$; 3) $\forall y \in Var_j - Var_i$ we have $l_j^y(s) = l_j^y(s')$; and 4) s' is reachable from s using transitions defined by T. Note that we discriminate the two accessibility relations by using $\sim_{i\to j}$ for CTL$^{cc,\alpha}$ and $\sim'_{i\to j}$ for CTLC^{+}. This definition shows that the accessibility relation $\sim'_{i\to j}$ is shift-reflexive and transitive, which is not the case in the CTL$^{cc,\alpha}$ model. The transitivity of the accessibility relation, as shown in Lemma 1 in [15], is indeed the reason of generating a spurious paradox, which we called fulfillment paradox and discussed in Chapter 1 along with a reasonable example. On the other hand, in the CTL$^{cc,\alpha}$ model, we added the condition ($(s, s') \in T$) in the definition of the social accessibility relation (Condition 2), which makes the accessibility relation not transitive so as to address this paradox. Figure 6 depicts an example of the CTLC^{+} model, which is not the CTL$^{cc,\alpha}$ model. By considering the four conditions of the accessibility relation $\sim'_{i\to j}$

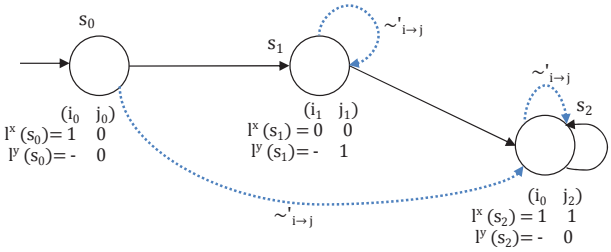

Figure 6: An example of the CTLC^{+} model, which is not the CTL$^{cc,\alpha}$ model

defined in [15]: 1) the local states of the debtor agent i are indistinguishable in s_0 and s_2 ($l_i(s_0) = l_i(s_2)$); 2) there exists one shared variable x and its value for the debtor i and creditor j in s_2 is the same ($l_i^x(s_0) = l_j^x(s_2)$); 3) the value of the unshared variable y for the creditor is unchanged in s_0 and s_2 ($l_j^y(s_0) = l_j^y(s_2)$); and 4) the accessible states are reachable using a sequence of transitions, we get s_2 is the only accessible state from s_0. It is obvious that this CTLC^{+} model cannot be considered as a CTL$^{cc,\alpha}$ model because s_2 should be the next state of s_0—not only reachable from s_0 using a sequence of transitions—to be an accessible state (Condition 2 in Definition 3.4.1). If we remove the accessibility from s_0 to s_2, the resulting model would not be neither CTLC^{+} nor CTL$^{cc,\alpha}$ because the seriality will be violated. Another example is shown in Figure 7. Part (a) depicts a CTL$^{cc,\alpha}$ model and part (b) shows the corresponding CTLC^{+} model. We can readily see that the two models are different.

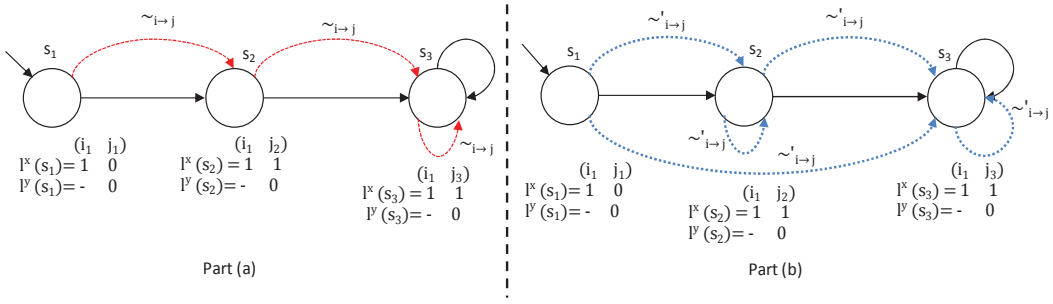

Figure 7: An example of the CTL$^{cc,\alpha}$ model (part (a)) and the CTLC^{+} model (part (b))

The model M is semantically unwind into a set of execution paths in which each path $\pi = s_0, s_1, \ldots$ is an infinite sequence of social states increasing simultaneously over time such that $s_i \in S$ and $(s_i, s_{i+1}) \in T$ for each $i \geq 0$. $\pi(k)$ is the k^{th} state of the path π. The set of all paths starting in s is denoted by $\Pi(s)$.

3.4.2 Syntax of CTL$^{cc,\alpha}$

Definition 3.4.2. *The syntax of CTL$^{cc,\alpha}$ is defined as follows:*

$$\varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid EX\varphi \mid EG\varphi \mid E(\varphi\, U\, \varphi) \mid \mathsf{CC} \mid \alpha$$

$$\mathsf{CC} ::= \xi CC(i, j, \varphi, \varphi)$$

$$\alpha ::= Fu\xi(i, \mathsf{CC}) \mid Ca\xi(i, \mathsf{CC}) \mid Re\xi(j, \mathsf{CC}) \mid De\xi(i, k, \mathsf{CC}) \mid As\xi(j, k, \mathsf{CC})$$

where:

- $\xi \in \{W, S\}$ *refers to weak and strong.* p, $\neg$, $\vee$, E, A, X, G, *and* U *are defined in Definition 2.1.1 in Chapter 2.*
- i *and* $j \in \mathcal{A}$ *are two agents.* WCC, SCC, FuW, FuS, CaW, CaS, ReW, ReS, DeW, DeS, AsW, *and* AsS *stand for weak and strong conditional commitments and their fulfillments, cancelations, releases, delegations, and assignments, respectively.*

The syntactic grammar rules of CTL$^{cc,\alpha}$ have in principle three different but integrated parts: propositional part, temporal part, and communication part.

Propositional part: The propositional part is in fact a propositional logic and consists of a set of atomic propositions and a set of Boolean connectives. Propositions are declarative statements that can be evaluated into true or false and represent essentially facts. Each fact is declared using the perfective aspect in the English language. For example, with respect to the NetBill protocol, atomic propositions will represent statements such as "the requested goods have been delivered", "the agreed payment has been made", and so on. Propositions with explicit time and propositions with explicit data, such as "the requested good is delivered by 5:30PM", and "the agreed payment is 500$" can also be approximated. Also, we denote the set of atomic propositions that hold at a given state s by $\mathcal{V}(s)$. We in turn use it to implement our methodology of representing declarative statements that include facts and facts with explicit domain variables (e.g., amount of money) as atomic propositions. Other Boolean connectives can be abbreviated in terms of the above as usual (see Definition 2.1.1 in Chapter 2).

Temporal part: The temporal part: 1) allows us to represent and reason about temporal qualitative requirements and to reason about the satisfaction of propositions in the past and future modes; and 2) uses the quantifiers to restrict the execution of paths. The reading of the formulae $EX\varphi$, $EG\varphi$, and $E(\varphi\, U\, \psi)$ and the abbreviations of other temporal operators are introduced in Definition 2.1.1 in Chapter 2.

Communication part: The communication part focuses on modeling interactions among agents using social commitments and their actions modalities. By using these modalities to express interaction requirements, the resulting properties are called communication properties. The formula $WCC(i, j, \psi, \varphi)$ (respectively, $SCC(i, j, \psi, \varphi)$) is read as "agent i weakly (respectively, strongly) commits towards agent j to consequently satisfy φ once the antecedent ψ holds". Since the antecedent ψ and the consequence φ in the context of commitment modality can be any arbitrary CTL$^{cc,\alpha}$ formula, so they would be conditional commitments as well. Commitment antecedents can also express the past using the until operator in the usual way (see for example our formalization to the scenario presented in Example 3.2.3). Moreover, as we mentioned earlier, the main difference between our two types of conditional commitments (weak and strong) is that a weak commitment can be activated even when the antecedent is false in all accessible states, while a strong commitment does exist solely when there exists at least one accessible state satisfying the antecedent. In other words, an agent can strongly commit only when there is a possibility that the antecedent could be satisfied given that the model is known at design time, thanks to the fact that the model has finite states. The insurance company's compliance with the commitment mentioned in Example 3.2.2 would be modeled as follows:

$$AG\Big(SCC(Ins, Pat, \big(A(\neg Claim(Reimbursed)\ U\ (Claim(Approved) \wedge \\ \neg\ Claim(Reimbursed))), EXEF\, Claim(Reimbursed)\big)\Big)$$

It means the insurance company always strongly commits to reimburse a covered patient's claim for a health procedure merely if the patient obtains an approval for her claim from the company prior to the health procedure; an antecedent which is possible to hold. The physician's compliance with the commitment mentioned in Example 3.2.3 is expressed as follows:

$$AG\Big(SCC(Phy, Pat, \big(\neg\ E(\neg HeartReport(Signed)\ U\ HeartReport(Trouble)), \\ \neg\ E(\neg Lab(Test)\ U\ Lab(Evaluate))\big)\Big)$$

It means the physician always strongly commits to a patient when the patient has any sign of heart trouble after signing up with her, then she will be immediately referred to a laboratory for tests in order to evaluate the results. This technical difference is cleared in Definition 3.4.3.

In the duplex party action formulae, $FuS(i, SCC(i, j, \psi, \varphi))$ (respectively, $FuW(i, WCC(i, j, \psi, \varphi))$) is read as "agent i has fulfilled her strong commitment $SCC(i, j, \psi, \varphi)$

(respectively, weak commitment $WCC(i, j, \psi, \varphi)$) towards agent j"; $CaS(i, SCC(i, j, \psi, \varphi))$ (respectively, $CaW(i, WCC(i, j, \psi, \varphi))$) is read as "agent i has canceled her strong commitment $SCC(i, j, \psi, \varphi)$ (respectively, weak commitment $WCC(i, j, \psi, \varphi)$) towards agent j"; and $ReS(j, SCC(i, j, \psi, \varphi))$ (respectively, $ReW(j, WCC(i, j, \psi, \varphi))$) is read as "agent j has released agent i from her strong commitment $SCC(i, j, \psi, \varphi)$ (respectively, weak commitment $WCC(i, j, \psi, \varphi)$)". In the triplex party action formulae, $DeS(i, k, SCC(i, j, \psi, \varphi))$ (respectively, $DeW(i, k, WCC(i, j, \psi, \varphi))$) is read as "agent i has delegated her role to agent k in her strong commitment $SCC(i, j, \psi, \varphi)$ (respectively, weak commitment $WCC(i, j, \psi, \varphi)$)"; and $AsS(j, k, SCC(i, j, \psi, \varphi))$ (respectively, $AsW(j, k, WCC(i, j, \psi, \varphi))$) is read as "agent j has assigned her strong commitment $SCC(i, j, \psi, \varphi)$ (respectively, weak commitment $WCC(i, j, \psi, \varphi)$) to agent k". For the readability purpose of the actions syntax, a redundant argument is added to represent the agent that has the right to perform the action under the assumption that the debtor agent creating a commitment can fulfill, cancel, and delegate the commitment, while the creditor agent can release, and assign the commitment directed to it.

Baldoni and colleagues [8, 9, 11] introduced a declarative language called 2CL to define constraints among commitments through a set of operators capturing patterns often used in interaction protocols. Internally, the constraints can be of different kinds: 1) some expressing temporal requirements on the satisfaction of antecedents and consequences; and 2) some other just capturing a relation among commitments. The grounded semantics of these constraints are defined by expressing their equivalent LTL formulae. Our language captures the temporal requirements between the antecedent and consequence of commitments, as shown above. Also, we can capture temporal requirements in an orthogonal direction about different commitments using the CTL$^{cc,\alpha}$ temporal operators (e.g., the U operator), while the relationship among different commitments can be defined using Boolean CTL$^{cc,\alpha}$ connectives. In terms of interacting agents' flexibility, everything that is not impressed by constraints is left free to the agents so as to define as propositions.

3.4.3 Semantics of CTL$^{cc,\alpha}$

Definition 3.4.3 (Satisfaction)**.** *Given the model M, the satisfaction of a CTL$^{cc,\alpha}$ formula φ in a social state s denoted by $(M, s) \models \varphi$ is defined in Tables 4, 5, 6, 7, 8, and 9.*

The formal semantics of the CTL formulae, a temporal fragment of CTL$^{cc,\alpha}$, is introduced in Chapter 2. In Table 4, the state formula $WCC(i, j, \psi, \varphi)$ is satisfied in the model M at s iff the consequence φ holds in every state satisfying ψ and accessible from s via

$\sim_{i\to j}$. The semantics of the strong commitment $SCC(i,j,\psi,\varphi)$ is similar, but we add Condition 1 to stress that the antecedent ψ and the consequent φ of commitments should be achieved at least in one state accessible from s, which is necessitated in business commitments in a strong manner. The proposed semantics is close to the semantics introduced

Table 4: The semantics of the weak and strong commitment formulae

$(M,s) \models WCC(i,j,\psi,\varphi)$	*iff*	$\forall s' \in S\ s.t.\ s \sim_{i\to j} s'$ *and* $(M,s') \models \psi$, *we have* $(M,s') \models \varphi$
$(M,s) \models SCC(i,j,\psi,\varphi)$	*iff*	1) $\exists s' \in S\ s.t.\ s \sim_{i\to j} s'$ *and* $(M,s') \models \psi$, *and* 2) $(M,s) \models WCC(i,j,\psi,\varphi)$

by Singh [81] for practical commitments. Singh's semantics is defined by first computing the set of states where the consequence holds and then testing whether these states are among the set of sets of states satisfying the antecedent, which in turn explicitly means this set of states should satisfy the consequence and antecedent of commitment together. Moreover, our semantics guarantee that any conflict between two or more strong commitments (enforceable commitments) will not exist as if one commitment has been activated, the other conflicting commitments will be not active in the same state (cf. R_{11} in Section 3.6). Also, two conflicting commitments can be individually enforceable in different states in the future.

In Table 5, the state formula $FuW(i, WCC(i,j,\psi,\varphi))$ is satisfied in the model M at s iff s satisfies the consequence φ and the negation of the weak commitment $WCC(i,j,\psi,\varphi)$ as well as there exists a state s' at which i performs an action to fulfill her commitment holding at s' and s is "seen" and reached from this state via $\sim_{i\to j}$ and R_{c_i}. The idea behind this semantics is to say that a weak commitment is fulfilled when we reach an

Table 5: The semantics of the fulfillment action formulae

$(M,s) \models FuW(i, WCC(i,j,\psi,\varphi))$	*iff* $\exists s' \in S\ s.t.\ s' \sim_{i\to j} s$ *and* $(l_i(s'), \mathit{Fulfill}_i, l_i(s)) \in R_{c_i}$ *and* $(M,s') \models WCC(i,j,\psi,\varphi)$ *and* $(M,s) \models \varphi \wedge \neg WCC(i,j,\psi,\varphi)$
$(M,s) \models FuS(i, SCC(i,j,\psi,\varphi))$	*iff* $\exists s' \in S\ s.t.\ s' \sim_{i\to j} s$ *and* $(l_i(s'), \mathit{Fulfill}_i, l_i(s)) \in R_{c_i}$ *and* $(M,s') \models SCC(i,j,\psi,\varphi)$ *and* $(M,s) \models \psi \wedge \neg SCC(i,j,\psi,\varphi)$

accessible state from the weak commitment state by performing the fulfillment action in which the consequence holds and the weak commitment becomes no longer active. The semantics of the strong fulfillment $FuS(i, SCC(i,j,\psi,\varphi))$ is similar, but the focus is on checking the satisfiability of the antecedent ψ. This is because—from the semantics of

the strong conditional commitment—we guarantee that whenever ψ holds in an accessible state, then the consequence φ holds as well. By stressing that the active commitment should be terminated in the fulfillment state, we address the fulfillment paradox appeared in [15] (Proposition 2) and discussed in Chapter 1. Recall that this paradox results from the assumption: unconditional commitment should be active when it comes time to its fulfillment, formally: $Fu(C(i,j,\varphi)) \rightarrow C(i,j,\varphi)$. Terminating (or deleting) commitment being fulfilled is stated explicitly in the operational semantics introduced in [106, 97, 81, 23]. Singh, for instance, uses the following postulate: $\varphi \rightarrow \neg\complement(i,j,\psi,\varphi)$ where $\complement$ could be a practical or dialectical conditional commitment [81].

In Table 6, the formula $CaS(i, SCC(i,j,\psi,\varphi))$ (respectively, $CaW(i, WCC(i,j,\psi,\varphi))$) is satisfied in M at s iff there exists a state s' at which i performs an action to cancel her strong (respectively, weak) commitment holding at s' and s is seen and reached from the state s' via $\sim_{i\rightarrow j}$ and R_{c_i}. And the current state s satisfies the negation of both the antecedent ψ (respectively, consequence φ) and strong commitment $SCC(i,j,\psi,\varphi)$ (respectively, weak commitment $WCC(i,j,\psi,\varphi)$).

Table 6: The semantics of the cancelation action formulae

$(M,s) \models CaS(i, SCC(i,j,\psi,\varphi))$	*iff* $\exists s' \in S$ *s.t.* $s' \sim_{i\rightarrow j} s$ *and* $(l_i(s'), Cancel_i, l_i(s)) \in R_{c_i}$ *and* $(M,s') \models SCC(i,j,\psi,\varphi)$ *and* $(M,s) \models \neg\psi \wedge \neg SCC(i,j,\psi,\varphi)$
$(M,s) \models CaW(i, WCC(i,j,\psi,\varphi))$	*iff* $\exists s' \in S$ *s.t.* $s' \sim_{i\rightarrow j} s$ *and* $(l_i(s'), Cancel_i, l_i(s)) \in R_{c_i}$ *and* $(M,s') \models WCC(i,j,\psi,\varphi)$ *and* $(M,s) \models \neg\varphi \wedge \neg WCC(i,j,\psi,\varphi)$

In Table 7, the formula $ReS(i, SCC(i,j,\psi,\varphi))$ (respectively, $ReW(i, WCC(i,j,\psi,\varphi))$) is satisfied in M at s iff there exists a state s' at which j performs an action to release the agent i from her strong (respectively, weak) commitment holding at s' and s is seen and reached from the state s' via $\sim_{i\rightarrow j}$ and R_{c_j}. And the current state s satisfies the negation of both the antecedent ψ (respectively, consequence φ) and strong commitment $SCC(i,j,\psi,\varphi)$ (respectively, weak commitment $WCC(i,j,\psi,\varphi)$). The only difference between the semantics of cancelation and release action formulae is the intelligent agent that performs these actions and the label of the local transition.

In Table 8, the formula $DeS(i,k, SCC(i,j,\psi,\varphi))$ (respectively, $DeW(i,k, WCC(i,j,\psi,\varphi))$) is satisfied in M at s iff there exists a state s' at which i performs an action to delegate her strong (respectively, weak) commitment holding at s' to another agent k and s is seen and reached from the state s' via $\sim_{i\rightarrow j}$ and R_{c_i}. And the current state s satisfies the negation of the antecedent ψ (respectively, consequence φ) and strong commitment

Table 7: The semantics of the release action formulae

$(M,s) \models ReS(j, SCC(i,j,\psi,\varphi))$	*iff* $\exists s' \in S$ *s.t.* $s' \sim_{i\to j} s$ *and* $(l_j(s'), Release_j, l_j(s)) \in R_{c_j}$ *and* $(M,s') \models SCC(i,j,\psi,\varphi)$ *and* $(M,s) \models \neg\psi \wedge \neg SCC(i,j,\psi,\varphi)$
$(M,s) \models ReW(j, WCC(i,j,\psi,\varphi))$	*iff* $\exists s' \in S$ *s.t.* $s' \sim_{i\to j} s$ *and* $(l_j(s'), Release_j, l_j(s)) \in R_{c_j}$ *and* $(M,s') \models WCC(i,j,\psi,\varphi)$ *and* $(M,s) \models \neg\psi \wedge \neg WCC(i,j,\psi,\varphi)$

$SCC(i,j,\psi,\varphi)$ (respectively, weak commitment $WCC(i,j,\psi,\varphi)$) as well as the new commitment created by k.

Table 8: The semantics of the delegation action formulae

$(M,s) \models DeS(i,k, SCC(i,j,\psi,\varphi))$	*iff* $\exists s' \in S$ *s.t.* $s' \sim_{i\to j} s$ *and* $(l_i(s'), Delegate_i, l_i(s)) \in R_{c_i}$ *and* $(M,s') \models SCC(i,j,\psi,\varphi)$ *and* $(M,s) \models \neg\psi \wedge \neg SCC(i,j,\psi,\varphi) \wedge SCC(k,j,\psi,\varphi)$
$(M,s) \models DeW(i,k, WCC(i,j,\psi,\varphi))$	*iff* $\exists s' \in S$ *s.t.* $s' \sim_{i\to j} s$ *and* $(l_i(s'), Delegate_i, l_i(s)) \in R_{c_i}$ *and* $(M,s') \models WCC(i,j,\psi,\varphi)$ *and* $(M,s) \models \neg\varphi \wedge \neg WCC(i,j,\psi,\varphi) \wedge WCC(k,j,\psi,\varphi)$

In Table 9, the formula $AsS(j,k,SCC(i,j,\psi,\varphi))$ (respectively, $AsW(j,k,WCC(i,j,\psi,\varphi))$) is satisfied in M at s iff there exists a state s' at which j performs an action to assign her strong (respectively, weak) commitment holding at s' to another agent k and s is seen and reached from the state s' via $\sim_{i\to j}$ and R_{c_j}. And the current state s satisfies the negation of the antecedent ψ (respectively, consequence φ) and strong commitment $SCC(i,j,\psi,\varphi)$ (respectively, weak commitment $WCC(i,j,\psi,\varphi)$) as well as the new commitment created by k. The differences between the semantics of delegation and assignment action

Table 9: The semantics of the assignment action formulae

$(M,s) \models AsS(j,k, SCC(i,j,\psi,\varphi))$	*iff* $\exists s' \in S$ *s.t.* $s' \sim_{i\to j} s$ *and* $(l_j(s'), Assign_j, l_j(s)) \in R_{c_j}$ *and* $(M,s') \models SCC(i,j,\psi,\varphi)$ *and* $(M,s) \models \neg\psi \wedge \neg SCC(i,j,\psi,\varphi) \wedge SCC(i,k,\psi,\varphi)$
$(M,s) \models AsW(j,k, WCC(i,j,\psi,\varphi))$	*iff* $\exists s' \in S$ *s.t.* $s' \sim_{i\to j} s$ *and* $(l_j(s'), Assign_j, l_j(s)) \in R_{c_j}$ *and* $(M,s') \models WCC(i,j,\psi,\varphi)$ *and* $(M,s) \models \neg\varphi \wedge \neg WCC(i,j,\psi,\varphi) \wedge WCC(i,k,\psi,\varphi)$

formulae are the agent that performs these actions and created new commitments. Simply

put, the idea of the proposed semantic rules is that a commitment is fulfilled (respectively cancelled, released, delegated and assigned) when we reach an accessible state from the commitment state by performing the fulfillment (respectively cancelation, release, delegation and assignment) action in which the antecedent holds (respectively not holds) and the commitment becomes no longer active as well as in the case of delegation and assignment action formulae, a new commitment is created.

3.5 Properties of CTL$^{cc,\alpha}$

Here, we present some properties of the developed CTL$^{cc,\alpha}$ logic. Such properties are valid (i.e., they are true in all states of all models). Specifically, we group these properties into three categories.

3.5.1 First group of properties

The former category captures the relationship between strong and weak commitments and their actions.

Proposition 3.5.1. $\models SCC(i,j,\psi,\varphi) \rightarrow WCC(i,j,\psi,\varphi)$

Proposition 3.5.2. $\models FuS(i,SCC(i,j,\psi,\varphi)) \rightarrow FuW(i,WCC(i,j,\psi,\varphi))$

Proposition 3.5.3. $\models SCC(i,j,\top,\varphi) \equiv WCC(i,j,\top,\varphi)$

Proposition 3.5.1 states that the set of strong commitments is a subset of the set of weak commitments. It intuitively means if an agent strongly commits to bring about something given an antecedent, it implicity weakly commits to bring about the same thing subject to the same antecedent. Consequently, fulfilling the strong commitment will automatically bring the fulfillment of the weak one (Proposition 3.5.2). From the defined semantics, we cannot generalize Proposition 3.5.2 to consider other actions. For example, we cannot say that when the strong commitment is canceled, then its weak commitment is canceled as well (see a counter-example in Figure 8). In this figure, the strong conditional commitment holding at s_0 can be canceled at s_2 (i.e., the formula $CaS(i,SCC(i,j,\psi,\varphi))$ holds at s_2) by performing the local cancel action (see the semantics of canceling strong commitments in Table 6). According to Proposition 3.5.1, the weak commitment $WCC(i,j,\psi,\varphi)$ holds as well at s_0, but it cannot be canceled at s_2 because its consequence φ is true at s_2. The intuition of Proposition 3.5.3 is that whenever the antecedent is true, the strong and weak

commitments became indistinguishable, since the antecedent doesn't play any role. The proof of these propositions is straightforward from the defined semantics.

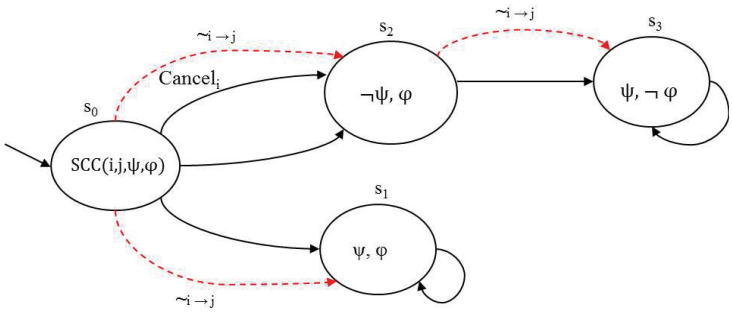

Figure 8: A model satisfies canceling strong commitment, but doesn't satisfy canceling weak one

3.5.2 Second group of properties

The second category investigates a special set of weak commitments, called pure-weak commitments, which are not strong commitments. To distinguish between pure-weak commitments and weak commitments, we use $PWCC(i, j, \cdot\ ,\ \cdot)$. Thus, the notation $WCC(i, j, \cdot\ ,\ \cdot)$ includes strong and pure cases $SCC(i, j, \cdot\ ,\ \cdot)$ and $PWCC(i, j, \cdot\ ,\ \cdot)$, respectively (cf. Figure 9). The semantics of pure-weak commitments is as follows:

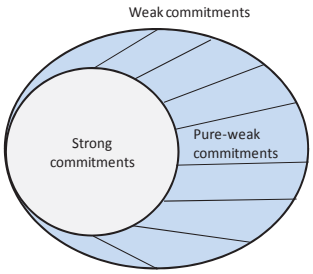

Figure 9: The relationship between weak, pure-weak, and strong commitments

$$(M, s) \models PWCC(i, j, \psi, \varphi) \ \textit{iff}\ \ (M, s) \models WCC(i, j, \psi, \varphi)\ \textit{and}\ (M, s) \not\models SCC(i, j, \psi, \varphi)$$

This semantic rule can be simplified as:

$$(M, s) \models PWCC(i, j\psi, \varphi)\ \textit{iff}\ \ \forall s' \in S\ s.t.\ s \sim_{i \to j} s',\ \textit{we have}\ (M, s') \models \neg\psi$$

Proposition 3.5.4. $\models PWCC(i,j,\psi,\varphi) \rightarrow WCC(i,j,\psi,\varphi)$

Proposition 3.5.5. $\models WCC(i,j,\psi,\varphi) \equiv SCC(i,j,\psi,\varphi) \vee PWCC(i,j,\psi,\varphi)$

Proposition 3.5.4 means that the set of pure-weak commitments is a subset of the set of weak commitments. It intuitively states that when an agent is pure-weakly committed (so the antecedent doesn't hold), then the agent is already weakly committed. Proposition 3.5.5 states that weak commitments could be either strong commitments or pure-weak commitments. The proof of these propositions is straightforward from the defined semantics of pure-weak, weak, and strong commitments.

3.5.3 Third group of properties

The last category captures the relation between conditional and unconditional commitments. There are two methods to define this relation. In the first method, we follow recent literatures (e.g., [81, 26]) that consider conditional commitment as a first class and treat unconditional commitment $C(i,j,\varphi)$[5] as a special case when the antecedent is always true. In our logic, we define unconditional commitment as abbreviation as follows: $C(i,j,\varphi) \equiv WCC(i,j,\top,\varphi)$.

In the second method, according to the defined semantics, one can define conditional commitments in terms of unconditional commitments as follows:

Proposition 3.5.6. $\models WCC(i,j,\psi,\varphi) \equiv C(i,j,\psi \rightarrow \varphi)$

Proposition 3.5.6 intuitively means that since weak conditional commitments can be created even if there is no possibility to satisfy their antecedents, then they are equivalent to the corresponding unconditional commitment using the material implication.

Proof. The proof is straightforward: Assume $(M,s) \models WCC(i,j,\psi,\varphi)$. From the semantics of the weak conditional commitment, for every $s' \in S$ such that $s \sim_{i \rightarrow j} s'$, if $(M,s') \models \psi$, we have $(M,s') \models \varphi$. Consequently, $(M,s') \models \psi \rightarrow \varphi$ for every $s' \in S$ such that $s \sim_{i \rightarrow j} s'$; so the equivalence. □

Proposition 3.5.7. $\models SCC(i,j,\psi,\varphi) \equiv C(i,j,\psi \rightarrow \varphi) \wedge \neg C(i,j,\neg\psi)$

Proposition 3.5.7 intuitively shows how strong conditional commitments can be defined using their corresponding unconditional commitments in a way similar to Proposition 3.5.6,

[5] For the sake of clarity, we use two different modalities to distinguish between the two types of commitments.

but with an extra condition that there is no commitment bringing about the negation of the antecedent. Such a condition ensures that there is at least one accessible state satisfying this antecedent. Specifically, the added formula $\neg C(i, j, \neg\psi)$ addresses the problem resulting from using the linear [76] and implication [50, 54] operators by stressing that there is a possibility to satisfy the commitment antecedent ψ. This formalization functions as the modeling of conditional commitments using the strict implication [79]: $C(i, j, \psi \rightsquigarrow \varphi)$ (see Chapter 2 for more details). In the case of unconditional commitments defined using weak commitments, the usage of material implication operator doesn't produce any technical problem because weak commitments can become active although their antecedents are always false. The semantics of weak commitments cannot be defined by the strict implication [79]. Moreover, from Prepositions 3.5.6 and 3.5.7, we obtain the following proposition:

Proposition 3.5.8. $\models SCC(i, j, \psi, \varphi) \equiv WCC(i, j, \psi, \varphi) \wedge \neg WCC(i, j, \top, \neg\psi)$

Proposition 3.5.8 intuitively means that when weak commitments hold and there is a possibility to satisfy their antecedents, then the corresponding strong ones hold as well.

Proof. Since $SCC(i, j, \psi, \varphi) \equiv C(i, j, \psi \rightarrow \varphi) \wedge \neg C(i, j, \neg\psi)$ and since $WCC(i, j, \psi, \varphi) \equiv C(i, j, \psi \rightarrow \varphi)$, we get $SCC(i, j, \psi, \varphi) \equiv WCC(i, j, \psi, \varphi) \wedge \neg C(i, j, \neg\psi)$ by using the substitution law. From
$WCC(i, j, \top, \varphi) \equiv C(i, j, \varphi)$, we obtain, $\neg WCC(i, j, \top, \neg\psi) \equiv \neg C(i, j, \neg\psi)$; so we are done. □

Since the two models of CTLC$^+$ and CTL$^{cc,\alpha}$ are different (see the discussion on Section 3.4), it is clear that the weak and strong conditional commitment modalities in CTL$^{cc,\alpha}$ cannot be defined in terms of the unconditional commitment modality $C(i, j, \cdot)$ in CTLC$^+$ introduced in [15]. To give a counterexample showing this fact, we use the model depicted in Figure 10 where the state s_1 is labeled by $WCC(i, j, \psi, \varphi)$ and $SCC(i, j, \psi, \varphi)$, but it is not labeled by $C(i, j, \psi \rightarrow \varphi)$, which is expressed in CTLC$^+$. This is because s_3, which is accessible from s_1 using the definition of the accessibility relation introduced in [15] satisfies $\psi \wedge \neg\varphi$. Thus, the equivalences presented in Propositions 3.5.6 and 3.5.7 don't hold for the unconditional commitment modality in CTLC$^+$. With this result, the unconditional commitment modality $C(i, j, \varphi)$ introduced in this thesis as the abbreviation of $WCC(i, j, \top, \varphi)$ is not equivalent to $C(i, j, \varphi)$ in CTLC$^+$.

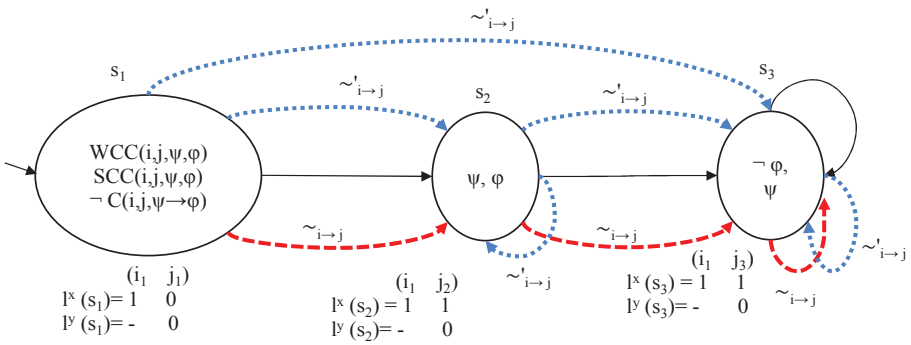

Figure 10: A model satisfies $WCC(i, j, \psi, \varphi)$ and $SCC(i, j, \psi, \varphi)$, but doesn't satisfy $C(i, j, \psi \rightarrow \varphi)$

3.6 Reasoning rules and action postulates

In this section, we present a set of reasoning rules and action postulates.

3.6.1 Reasoning rules about CTL$^{cc,\alpha}$

In addition to reasoning about conditional commitments using conditional commitment actions, we consider several reasoning rules along with proofs that they are supported in our logic. The motivation behind these rules is to: 1) capture the semantic characteristics of commitments and their actions; 2) show how two types of commitments and their actions are related to each other; 3) accommodate a generic way to deal with exceptions to achieve consistency; and 4) show how such rules are discussed in the literature of operational semantics for social commitments. As far as the debtor i and creditor j are understood from the context, we simply write $WCC(\psi, \varphi)$ instead of $WCC(i, j, \psi, \varphi)$ and $C(\varphi)$ instead of $C(i, j, \varphi)$. We note that when the rule holds for weak commitments, which means for pure-weak or strong cases (cf. Proposition 3.5.5), then it is enough to provide the proof for the general weak case. However, in the rest, we sometimes provide the proof for the strong case to better show the specificity of that case.

R$_1$ FULFILLMENT NECESSITY.

Formalization. $FuW(i, WCC(\psi, \varphi)) \rightarrow \varphi$.

Meaning. When a weak commitment is fulfilled, its consequence holds.

The proof is straightforward from the semantics of the fulfillment of weak commitments. This rule is very reasonable and natural in our logical model. R_1 is incorporated in the axioms of fulfillment introduced by Bentahar et al. [15]. However, Bentahar et al.'s validity is mainly related to unconditional commitments. A similar rule is also incorporated in [23, 106]. Analogous to the rule R_1, once a weak commitment is canceled, released, delegated or assigned, the commitment consequence is no longer active. The proof is also direct from the defined semantics.

R_2 FULFILLMENT.

Formalization. $FuW(i, WCC(\psi, \varphi)) \rightarrow \neg WCC(\psi, \varphi)$.

Meaning. The commitment is discharged and no longer active once it is fulfilled.

The proof is straightforward from the semantics of the two fulfillment modal operators. It is worth noticing that R_2 reveals our solution to the fulfillment paradox appeared in [15, 52]. Chopra and Singh [24], Yolum and Singh [106], Winikoff et al. [97], and Singh [81] incorporate this rule. For instance, Singh [81] defined R_2 as follows: $\varphi \rightarrow \neg CC^w(\psi, \varphi)$ where $w \in \{p, d\}$ such that p refers to practical commitments (commitments about what has to be done) while d refers to dialogical commitments (commitments about what holds). Since Singh's logical model includes solely commitment modalities, his semantics of the discharge (fulfillment) action focuses on checking the truth condition of the commitment consequent like in R_2 instead of considering a fulfillment modality. However, our rule and Singh's rule still have the same role, function, and intuition. Moreover, Chesani et al. [23] have considered this rule in their axiomatization about modeling unconditional commitment discharge using *event calculus*. Bentahar et al. [15] and El Menshawy et al. [52] don't incorporate this rule when defining semantics of fulfilling unconditional commitments. Analogous to the rule R_2, once a weak commitment is canceled, released, delegated or assigned, the commitment is no longer active. The proof is also direct from the defined semantics.

R_3 PARTIALLY DETACH.

Formalization. $WCC(\psi_1 \wedge \psi_2, \varphi) \wedge AX\psi_1 \rightarrow WCC(\psi_2, \varphi)$.

Meaning. When part of the antecedent (i.e., ψ_1) of a commitment holds in the next state, the commitment with the remainder of the antecedent (i.e., ψ_2) and the same consequence comes into being.

Proof. Let us consider the case of strong commitments; the case of pure-weak commitments is quite similar. Assume $(M, s) \models SCC(\psi_1 \wedge \psi_2, \varphi) \wedge AX\psi_1$. From the semantics of $SCC(\psi_1 \wedge \psi_2, \varphi)$, there exists a state $s' \in S$ such that $s \sim_{i \to j} s'$ and $(M, s') \models \psi_1 \wedge \psi_2$ (i.e., $(M, s') \models \psi_1$ and $(M, s') \models \psi_2$) and $(M, s') \models \varphi$ for all $s' \in S$ such that $s \sim_{i \to j} s'$ and $(M, s') \models \psi_1 \wedge \psi_2$. From condition 2 of the definition of the accessibility relation $s \sim_{i \to j} s'$ (i.e., $(s, s') \in T$) and the truth condition of "$AX\psi_1$" holding at the current state s, we conclude that for all accessible states s' from s, we have $(M, s') \models \psi_1$. Consequently, we cannot find an accessible state s' from s such that $(M, s') \models \psi_2 \wedge \neg\varphi$, because this would mean $(M, s') \models \psi_1 \wedge \psi_2 \wedge \neg\varphi$, which is contradictory with the above. Since there exists $(M, s') \models \psi_2$, we are done. □

For example, $WCC(Pha, Pat, Prescription(Showed) \wedge Prescription(Paid),$ $EF\ Medicine(Delivered))) \wedge AXPrescription(Showed)) \rightarrow WCC(Pha, Pat,$ $Prescription(Paid), EF\ Medicine\ (Delivered)))$, meaning that if a pharmacy weakly commits to eventually deliver medicines when a patient pays and shows a prescription, then as soon as the patient shows the prescription, the pharmacy can be weakly committed to eventually deliver medicines if the patient pays. Since in our model committing is an action that takes time (which makes our accessible states to be next states), we have to add "AX" in front of ψ_1. However, in Singh [81], committing is instantaneous and thus R_3 is defined as follows: $CC^w(\psi_1 \wedge \psi_2, \varphi) \wedge \psi_1 \rightarrow CC^w(\psi_2, \varphi)$ where $w \in \{p, d\}$. Technically, the difference with our rule is due to the fact that Singh [81] doesn't use the accessibility relation as in standard modal logic; instead he uses Segerberg's idea to define the semantics of the conditional commitment modalities.

R_4 FULLY DETACH.

Formalization. $WCC(\psi, \varphi) \wedge AX\psi \rightarrow C(\varphi)$.

Meaning. As a special case of R_3, we can detach a commitment into the corresponding unconditional commitment in one shot when its antecedent is satisfied.

For example, suppose $\psi = Prescription(Paid) \wedge Prescription(Showed)$ and $\varphi = EF\ Medicine(Delivered)$, then once the patient in one time pays and shows a prescription and ψ holds in all next states, the commitment $WCC(Pha, Pat, \psi, \varphi)$ is transformed into $C(Pha, Pat, \varphi)$. R_4 is supported by Yolum and Singh [106], Chopra and Singh [24], Winikoff et al. [97], and by Singh [81]. Furthermore, Chesani et al. [23] introduced two axioms for detaching a conditional commitment:

as soon as ψ holds, 1) the conditional commitment is terminated; and 2) the unconditional commitment is initiated.

R$_5$ L-DISJOIN.

Formalization. $WCC(\psi_1, \varphi) \wedge WCC(\psi_2, \varphi) \rightarrow WCC(\psi_1 \vee \psi_2, \varphi)$

Meaning. If i commits that φ if ψ_1 and commits that the same consequence holds if ψ_2, then i commits that φ if ψ_1 or ψ_2.

Proof. Here again we prove only the case of strong commitments (i.e., W=S) and the proof of pure-weak commitments is similar. Assume $(M, s) \models SCC(\psi_1, \varphi) \wedge SCC(\psi_2, \varphi)$. Therefore, there exists $s' \in S$ such that $s \sim_{i \rightarrow j} s'$ and $(M, s') \models \psi_1 \vee \psi_2$ and $(M, s') \models \varphi$ for all $s' \in S$ such that $s \sim_{i \rightarrow j} s'$ and $(M, s') \models \psi_1 \vee \psi_2$; so the rule. □

R$_6$ R-CONJOIN.

Formalization. $WCC(\psi, \varphi_1) \wedge WCC(\psi, \varphi_2) \rightarrow WCC(\psi, \varphi_1 \wedge \varphi_2)$

Meaning. When ψ holds, an agent i would become committed to bring about φ_1 and φ_2 if i double commits to bring about φ_1 if ψ and to bring about φ_2 if the same condition holds.

The proof is direct from the semantics. Suppose, for instance, $WCC(Mer, Cus, Good(Paid), EFGood(Delivered)) \wedge WCC(Mer, Cus, Good(Paid), EFWarranty(Delivered)))$, then the merchant weakly commits to the customer to eventually deliver goods and eventually deliver warranty paperwork if the customer sends the payment: $WCC(Mer, Cus, Good(Paid), EFGood(Delivered) \wedge EFWarranty(Delivered)))$.

Singh [81] presents a rule called a monotonicity as follows: "From $CC^w(\psi_1, \varphi)$, $\psi_2 \vdash \psi_1$ infer $CC^w(\psi_2, \varphi)$" where $w \in \{p, d\}$ and $\psi_2 \vdash \psi_1$ means ψ_1 is provable from ψ_2. The generalization indicates that any commitment that holds for a weaker antecedent also holds for a stronger one. Our monotonicity rule is defined as follows with respect to Singh's inference rule ($\vdash$):

R$_7$ MONOTONICITY.

Formalization. 1. $PWCC(\psi_1 \vee \psi_2, \varphi) \rightarrow PWCC(\psi_1, \varphi)$

2. $PWCC(\psi_1, \varphi) \rightarrow PWCC(\psi_1 \wedge \psi_2, \varphi)$

3. $C(\varphi) \rightarrow WCC(\psi, \varphi)$

Meaning. 1) an agent becomes pure-weakly committed to φ subject to ψ_1 if it pure-weakly commits to the same consequence subject to $\psi_1 \vee \psi_2$; 2) an agent pure-weakly commits to φ subject to $\psi_1 \wedge \psi_2$ if it pure-weakly commits to the same consequence subject to only one of the two parts of the conjunction; and 3) an agent becomes weakly committed to φ subject to ψ if it already commits to φ.

Proof. From the previous proofs and semantics of the pure-weak and weak commitment modalities, the proof of the three forms of monotonicity is direct. We merely present the proof of the third monotonicity rule. Assume $(M, s) \models C(i, j, \varphi)$. Therefore, for every $s' \in S$ such that $s \sim_{i \to j} s'$, we have $(M, s') \models \varphi$; consequently, there is no accessible state that satisfies $\psi \wedge \neg\varphi$ (for whatever ψ). As a result, for every $s' \in S$ such that $s \sim_{i \to j} s'$ and $(M, s') \models \psi$, we have $(M, s') \models \varphi$; so we are done. □

For strong commitments, the monotonicity rules need additional conditions: for property (1), $\psi_1 = \top$ should be added in the antecedent; for property (2), the antecedent should include $\psi_2 = \top$; and for property (3), ψ should be true. The reason for these additional requirements with regard to strong commitments is because the antecedent of these commitments cannot be false (this will be clear in R_8). Furthermore, the property (3) captures the relationship between conditional and unconditional commitments when the antecedent is always true.

R_8 CONSISTENCY (Strong Commitment).

Formalization. $\neg SCC(\psi, \bot)$, where $\bot$ is read as "constant false proposition" and is abbreviated as $\bot \equiv \neg\top$

Meaning. An agent cannot strongly commit to false.

The proof is straightforward from the semantics of the strong commitment operator. This rule is valid only when the conditional commitment in question is strong. This is because the semantics of such commitment requires the existence of an accessibility state that satisfies the antecedent ψ, which is not the case of pure-weak commitments. Consequently, the following holds: $PWCC(\bot, \bot)$, or in general $PWCC(\bot, \varphi)$ because there is no accessible state that satisfies $\bot$. However, this commitment will never be satisfied because the third condition in the fulfillment semantics, which says that the model satisfies $\neg PWCC(\bot, \varphi)$, will never happen. R_8 is also integrated in Singh's postulates (Postulate B_6 in [81]).

R_9 CONSISTENCY (Weak Commitment).

Formalization. $AX\psi \rightarrow \neg WCC(\psi, \bot)$

Meaning. An agent cannot weakly commitment to false once the commitment antecedent has been achieved.

Proof. From the semantics of $AX\psi$ and the second condition of the accessibility relation, it follows that all accessible states satisfying ψ. Since no accessible state satisfies $\bot$, the rule follows. □

$\mathbf{R}_{10}$ CONSISTENCY (Fulfilment).

Formalization. $\neg FuW(i, WCC(\psi, \bot))$

Meaning. A commitment to false cannot be fulfilled.

Proof. The proof is direct from the semantics of the fulfillment operator since there is no accessible state s such that $(M, s) \models \bot$. □

$\mathbf{R}_{11}$ STRONG CONSISTENCY.

Formalization. $SCC(\psi, \varphi) \rightarrow \neg SCC(\psi, \neg\varphi)$

Meaning. When a strong commitment holds, then there is no possibility for committing to the negation of its consequence subject to the same antecedent.

Proof. The rule follows directly from the fact that the semantics of strong commitments requires the existence of an accessible state satisfying ψ and a state satisfying ψ cannot satisfy φ and $\neg\varphi$. □

This rule is also incorporated in [81], and it is worth noticing that such a rule is not valid in the case of pure-weak commitments since $PWCC(\bot, \varphi)$ holds. It is stronger than R_6.

$\mathbf{R}_{12}$ WEAK CONSISTENCY.

Formalization. $AX\psi \wedge WCC(\psi, \varphi) \rightarrow \neg WCC(\psi, \neg\varphi)$

Meaning. When a weak commitment and its antecedent hold, then there is no possibility for committing to the negation of its consequence subject to the same antecedent.

The proof comes directly from the semantics of weak commitments and $AX\psi$.

$\mathbf{R}_{13}$ CHAIN.

Formalization. $(WCC(\psi_1, \varphi_1) \wedge AX(\varphi_1 \rightarrow \psi_2) \wedge WCC(\psi_2, \varphi_2)) \rightarrow WCC(\psi_1, \varphi_2)$

Meaning. Commitments are close under implication.

Proof. From the first commitment, in all accessible states that satisfy ψ_1, φ_1 holds, and by modus ponens, ψ_2 holds. Thus, from the second commitment, φ_2 holds; so the commitment on the right side holds. □

We can generalize the rule as follows: "From $WCC(\psi_1, \varphi_1), \varphi_1 \vdash \psi_2, WCC(\psi_2, \varphi_2)$ infer $WCC(\psi_1, \varphi_2)$".

R_{14} WEAKEN.

Formalization. $WCC(\psi, \varphi_1 \wedge \varphi_2) \rightarrow WCC(\psi, \varphi_1)$

Meaning. If i commits to a conjunction subject to a antecedent, i is also committed to each part of the conjunction.

Proof. If no accessible state satisfies the antecedent ψ, then both sides of the rule hold for the case of weak commitments. Otherwise, any accessible state that satisfies $\varphi_1 \wedge \varphi_2$ also satisfies φ_1. □

Singh [81] incorporates this rule by requiring $\neg\varphi_1$ on the left side of the rule, so that this rule is consistent with his DISCHARGE rule. Chopra and Singh [24] and Winikoff et al. [97] in their operationalization of commitments forcefully claimed that once a commitment is detached (i.e., its antecedent holds), we don't need to reason about it further. Singh [81] argued that although this rule is unnatural, it can be captured by reasoning on the unconditional commitments and "there is no need to state the conditional one although it continues to hold". In our framework, we can capture the detach in the same way since R_4 yields $C(\varphi)$. Chesani et al. [23] use a similar technique. Furthermore, the following rule is also captured:

R_{15} NONEXISTENCE.

Formalization. $AX(\psi \wedge \neg\varphi) \rightarrow \neg WCC(\psi, \varphi)$

Meaning. If the antecedent is brought about but the consequence also doesn't hold, then the commitment also doesn't hold.

The proof is straightforward from the semantics of the next and commitment operators and the second condition of the accessibility relation.

To summarize, our model supports most of the reasoning rules agreed upon in the literature about commitments and their actions that agents are expected to respect when they communicate.

3.6.2 CTL$^{cc,\alpha}$ action postulates

Like Singh [81], we augment the proposed semantics by additional rules under the form of postulates. The purpose is to force the satisfaction of business rules, which are intuitive, sound, and reasonable in commitments, commitment actions, and business protocols under the following constraints: 1) for every conditional commitment, when a software agent performed a commitment action, other commitment actions cannot be applied in all states for every paths starting from a state satisfying the corresponding commitment action formula ($Cond_1$); and 2) for every conditional commitment, only a software agent can perform a commitment action on a given commitment state ($Cond_2$).

To guarantee the satisfaction of our postulates, $Cond_1$ and $Cond_2$ should be considered in the models of MASs. Let $\xi \in \{W, S\}$ along with $Cond_1$ and $Cond_2$ be true, then our postulates have the following three patterns:

Future computation patterns

1. $Fu\xi(i, \xi CC(i, j, \psi, \varphi)) \rightarrow AXAG\neg Ca\xi(i, \xi CC(i, j, \psi, \varphi))$
2. $Fu\xi(i, \xi CC(i, j, \psi, \varphi)) \rightarrow AXAG\neg Re\xi(j, \xi CC(i, j, \psi, \varphi))$
3. $Fu\xi(i, (\xi CC(i, j, \psi, \varphi)) \rightarrow AXAG\neg De\xi(i, k, \xi CC(i, j, \psi, \varphi))$
4. $Fu\xi(i, (\xi CC(i, j, \psi, \varphi)) \rightarrow AXAG\neg As\xi(j, k, \xi CC(i, j, \psi, \varphi))$
5. $Ca\xi(i, (\xi CC(i, j, \psi, \varphi)) \rightarrow AXAG\neg Fu\xi(i, \xi CC(i, j, \psi, \varphi))$
6. $Re\xi(i, (\xi CC(i, j, \psi, \varphi)) \rightarrow AXAG\neg Fu\xi(i, \xi CC(i, j, \psi, \varphi))$
7. $De\xi(i, k, (\xi CC(i, j, \psi, \varphi)) \rightarrow AXAG\neg Fu\xi(i, \xi CC(i, j, \psi, \varphi))$
8. $As\xi(j, k, (\xi CC(i, j, \psi, \varphi)) \rightarrow AXAG\neg Fu\xi(i, \xi CC(i, j, \psi, \varphi))$

The first four patterns state that once fulfilled, strong and weak conditional commitments cannot be canceled, released, delegated or assigned again in the future computations. Other patterns can be read in a similar way.

Immediate state patterns

1. $Fu\xi(i, \xi CC(i, j, \psi, \varphi)) \rightarrow \neg Ca\xi(i, \xi CC(i, j, \psi, \varphi))$
2. $Fu\xi(i, \xi CC(i, j, \psi, \varphi)) \rightarrow \neg Re\xi(j, \xi CC(i, j, \psi, \varphi))$
3. $Fu\xi(i, \xi CC(i, j, \psi, \varphi)) \rightarrow \neg De\xi(i, k, \xi CC(i, j, \psi, \varphi))$
4. $Fu\xi(i, \xi CC(i, j, \psi, \varphi)) \rightarrow \neg As\xi(j, k, \xi CC(i, j, \psi, \varphi))$

While the previous patterns emphasize the consequence of the fulfillment action from the next state, the present patterns focus on the impact of actions on the current state. For example, when the strong and weak conditional commitment are fulfilled, then there is no possibility to apply cancel, release, delegate or assign action on the same commitment in the same state, which is reasonable, and intuitive. Other patterns can be read in a similar way.

Undesirable pattern

The undesirable pattern comes out when there is no possibility to fulfill, cancel, release, delegate and assign the activated conditional commitment in all states of every possible path. Formally, this pattern is specified as follows:

$$\neg\Big[EF\xi CC(i,j,\psi,\varphi) \wedge AG\Big((\neg Fu\xi(i,\xi CC(i,j,\psi,\varphi)) \wedge \neg Ca\xi(i,\xi CC(i,j,\psi,\varphi)) \wedge \neg Re\xi(j,\xi CC(i,j,\psi,\varphi)) \wedge \neg De\xi(i,k,\xi CC(i,j,\psi,\varphi)) \wedge \neg As\xi(j,k,\xi CC(i,j,\psi,\varphi)))\Big)\Big]$$

We do not claim that the set of our postulates is complete, but we can investigate the completeness[6] of any subset of our logic that supports the postulates using the corresponding theory following the methodology introduced by Singh [81]. This means other patterns can be added (e.g., if a commitment is canceled, it cannot be delegated in the future).

[6]The completeness is beyond the scope of this thesis and it is one of the main direction of future work.

Chapter 4

Symbolic Model Checking for CTL$^{cc,\alpha}$ and Implementation

This chapter[1] covers:

- An overview about the model checking technique and our symbolic verification technique.
- The development of new symbolic model checking algorithm dedicated to CTL$^{cc,\alpha}$, which addresses the first part of Question 1.3.6 mentioned in Chapter 1.
- Theoretical results about our algorithm.
- The implementation of our symbolic model checking algorithm and tool, which addresses the second part of Question 1.3.6 mentioned in Chapter 1.

4.1 Overview

Since it is not known in advance whether a party will satisfy its commitment, then verifying whether or not the commitment is resolved is of a significant importance, especially agents are autonomous and heterogeneous. The term 'resolved' refers to either fulfill, cancel, release, delegate, or assign action. Testing and simulation are the most widely employed verification techniques [32]. They are carried out at run time to enable designers to test design models or certain conditions, which are difficult or expensive to reproduce in the real world. When the systems (e.g., MASs) under the consideration of verification process have a complex and large state-space and their components need to synchronously or

[1]The results in this chapter are collected from our publications in [39, 43, 44].

asynchronously communicate with each other, these techniques cannot guarantee the full coverage of all possible behaviors [32]. Model checking, performed at design time, is another verification technique that can complement testing and simulation [32, 33, 72]. In the model checking technique, there are two main steps. In the first step, a real-world system is encoded as a logical model M to represent key aspects of the system, including the fundamental requirements, components, and how those components communicate with each other. In the second step, a set of requirements (or specifications) are expressed in a temporal logic as logical formulae, named φ_1, φ_2, etc. By thoroughly exploring the run-time states (also called state-space) of the system model without running the system, it is then possible to automate the verification of whether $M \models \varphi_1$ (or $M \not\models \varphi_1$), i.e., whether the system model M satisfies (or doesn't satisfy) the formula φ_1. The model checking termination is assured by the finiteness of the system model [32, 33] and this is why the model checking technique can perform "exhaustive analysis".

In this chapter, we investigate the problem of model checking CTL$^{cc,\alpha}$ in a practical way. Given a model M representing a MAS and a CTL$^{cc,\alpha}$ formula φ expressing a property, the decision problem of model checking CTL$^{cc,\alpha}$ is determining whether or not M is a model for φ. It would be formalized as: $(M, I) \models \varphi$ *iff* $(M, s) \models \varphi \; \forall s \in I$ where I is the set of initial states. We adopt symbolic model checking techniques to tackle this problem for several technical reasons mentioned in Chapter 1. In principle, a symbolic model checker differs from a symbolic model checking algorithm, although they have the same inputs: Encoded model and set of formulae. Specifically, a symbolic model checking algorithm computes the set of states $[\![\varphi]\!]$ in the model M satisfying the given formula φ. The set $[\![\varphi]\!]$ would be formalized as $[\![\varphi]\!] = \{s \in S \mid (M, s) \models \varphi\}$. On the other hand, a symbolic model checker is a tool that is built on top of a symbolic model checking algorithm. One of the most powerful features of this tool is the production of counter-examples (i.e., witnesses of the offending system behaviors) when the specifications are violated [32].

Figure 11 shows our symbolic verification technique. Algorithm 1 reports the general structure and function of our symbolic model checker for automatically checking the satisfaction of CTL$^{cc,\alpha}$ formulae. Since our symbolic model checker is the extension of the symbolic model checker MCMAS [64] with our symbolic model checking algorithm, we call it MCMAS+. It starts with comparing the set of initial states I against the set $[\![\varphi]\!]$ of states that satisfy CTL$^{cc,\alpha}$ formula φ. This set is computed by our symbolic model checking algorithm (refer to Algorithm 2). If $I \subseteq [\![\varphi]\!]$, it returns true and witness-example; otherwise

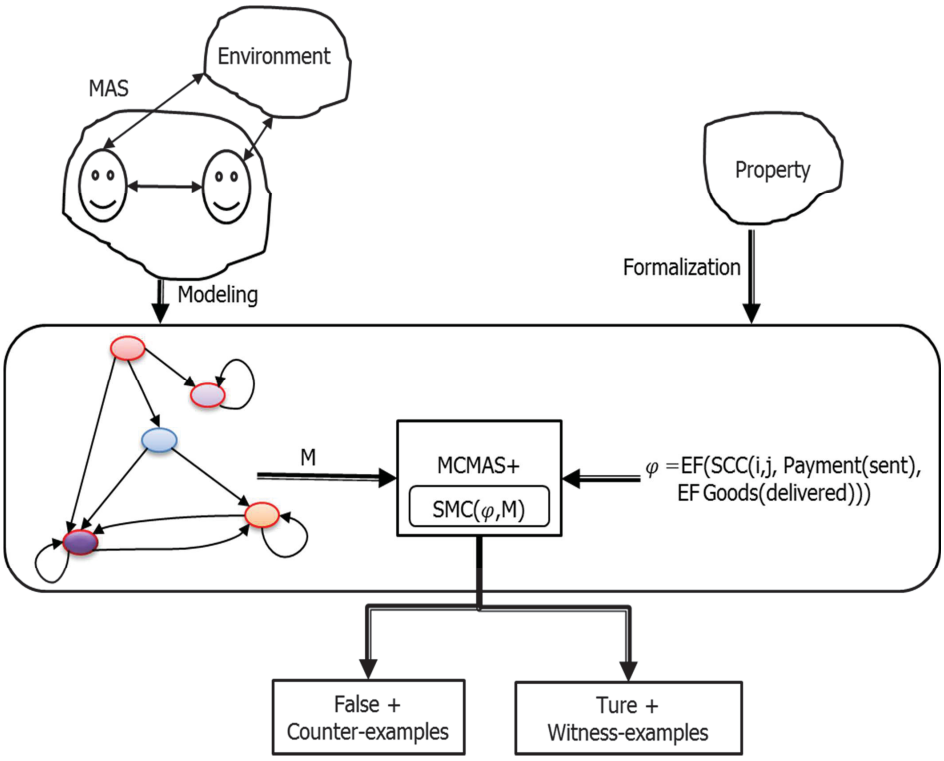

Figure 11: Our symbolic verification technique

it returns false and counter-example. Notably, producing counter-examples (respectively,

Algorithm 1 MCMAS+(M, φ): Boolean

1: **if** $I \subseteq [\![\varphi]\!]$ **then**
2: return TRUE and witness-example
3: **else**
4: return FALSE and counter-example
5: **end if**

witness-examples) for some kind of formulae such as EFp (respectively, $AG\neg p$) needs to generate the whole model. Other model checkers such as CWB-NC[2] do not produce counter-examples.

4.2 Symbolic model checking algorithm for CTL$^{cc,\alpha}$

In principle, our symbolic model checking algorithm extends the standard CTL symbolic algorithm introduced in [32]. Our main algorithm described in Algorithm 2 accepts the model M and the CTL$^{cc,\alpha}$ formula φ as input and brings back the set $[\![\varphi]\!]$ of states that

[2] `http://www.cs.sunysb.edu/cwb/`.

Algorithm 2 $SMC(\varphi, M)$: the set $[\![\varphi]\!]$ of states satisfying the CTL$^{cc,\alpha}$ formula φ

1: φ is an atomic formula: return $\mathcal{V}(\varphi)$;
2: φ is $\neg\varphi_1$: return S–$SMC(\varphi_1, M)$;
3: φ is $\varphi_1 \vee \varphi_2$: return $SMC(\varphi_1, M) \cup SMC(\ \varphi_2, M)$;
4: φ is $EX\varphi_1$: return $SMC_{EX}(\varphi_1, M)$;
5: φ is $E(\varphi_1\ U\ \varphi_2)$: return $SMC_{EU}(\varphi_1, \varphi_2, M)$;
6: φ is $EG\varphi_1$: return $SMC_{EG}(\varphi_1, M)$;
7: φ is $WCC(i, j, \varphi_1, \varphi_2)$: return $SMC_{wcc}(i, j, \varphi_1, \varphi_2, M)$;
8: φ is $SCC(i, j, \varphi_1, \varphi_2)$: return $SMC_{scc}(i, j, \varphi_1, \varphi_2, M)$;
9: φ is $FuW(i, WCC(i, j, \varphi_1, \varphi_2))$: return $SMC_{fuw}(i, j, \varphi_1, \varphi_2, M)$;
10: φ is $FuS(i, SCC(i, j, \varphi_1, \varphi_2))$: return $SMC_{fus}(i, j, \varphi_1, \varphi_2, M)$;
11: φ is $CaW(i, WCC(i, j, \varphi_1, \varphi_2))$: return $SMC_{caw}(i, j, \varphi_1, \varphi_2, M)$;
12: φ is $CaS(i, SCC(i, j, \varphi_1, \varphi_2))$: return $SMC_{cas}(i, j, \varphi_1, \varphi_2, M)$;
13: φ is $ReW(j, WCC(i, j, \varphi_1, \varphi_2))$: return $SMC_{rew}(i, j, \varphi_1, \varphi_2, M)$;
14: φ is $ReS(j, SCC(i, j, \varphi_1, \varphi_2))$: return $SMC_{res}(i, j, \varphi_1, \varphi_2, M)$;
15: φ is $DeW(i, k, WCC(i, j, \varphi_1, \varphi_2))$: return $SMC_{dew}(i, j, k, \varphi_1, \varphi_2, M)$;
16: φ is $DeS(i, k, SCC(i, j, \varphi_1, \varphi_2))$: return $SMC_{des}(i, j, k, \varphi_1, \varphi_2, M)$;
17: φ is $AsW(j, k, WCC(i, j, \varphi_1, \varphi_2))$: return $SMC_{asw}(i, j, k, \varphi_1, \varphi_2, M)$;
18: φ is $AsS(j, k, SCC(i, j, \varphi_1, \varphi_2))$: return $SMC_{ass}(i, j, k, \varphi_1, \varphi_2, M)$.

satisfy φ in M. Given the model M, the algorithm recursively runs on the structure of φ and constructs the set $[\![\varphi]\!]$ with respect to a set of Boolean operations applied to sets (e.g., *complementation, union*, and *existential quantification* operations). The standard CTL algorithms are called in lines from 1 to 6 to compute the set of states that satisfies pure CTL formulae. The algorithm then goes forward to invoke our sub-algorithms in lines from 7 to 18. The later algorithms calculate the set of states where our new modalities are true.

4.2.1 Two auxiliary algorithms

With respect to the modularity principal, we develop two auxiliary algorithms, which are needed in the computation of our sub-algorithms. The first algorithm $SMC_{\sim}(i, j, s', M)$ reported in Algorithm 3 is developed to calculate the set $Pre_{\sim}(s')$ of preimages which can

Algorithm 3 $SMC_{\sim}(i, j, s', M)$: the set $Pre_{\sim}(s')$

1: $X \leftarrow \Big\{ s \in S \mid l_i(s) = l_i(s')$ and $(s, s') \in T$ and $Var_i \cap Var_j \neq \emptyset$ and $l_i^y(s) = l_j^y(s')\ \forall y \in Var_i \cap Var_j$ and $l_j^z(s) = l_j^z(s')\ \forall z \in Var_j - Var_i \Big\}$;
2: return X;

see s' through the accessibility relation $\sim_{i\rightarrow j}$. Formally, $Pre_{\sim}(s') = \{s \in S \mid s \sim_{i\rightarrow j} s'\}$. The soundness of this algorithm is straightforward from the definition of $\sim_{i\rightarrow j}$. In a similar way, the second algorithm $SMC_{R_{ci}}(i, j, a_i, s', M)$ described in Algorithm 4 is developed to compute the set $Pre_{R_{ci}}(s')$ of preimages which can reach s' via the local transition. This transition is computed by R_{c_i} and labeled with a commitment action a_i. Formally, $Pre_{R_{ci}}(s') = \{s \in S \mid (s, a_i, s') \in R_{c_i}\}$.

Algorithm 4 $SMC_{R_{ci}}(i, j, a_i, s', M)$: the set $Pre_{R_{ci}}(s')$

1: $X \leftarrow \{s \in S \mid (l_i(s), a_i, l_i(s')) \in R_{c_i}\}$;
2: return X;

4.2.2 Running model

The business process model illustrated in Figure 12 is considered to clarify the computations of the developed algorithms. The social accessibility relations are represented by the dashed arrows. The values of the shared and unshared variables at each state are given and the labeled transitions are defined by R_{c_ξ} where $\xi \in \{i, k, m\}$. More precisely, the model captures the contractual relationships between two organizations. The first organization (e.g., Concordia University) is represented by one agent called customer (j) and the second organization (e.g., IBM company) consists in four departments: Customer service (i), information technology (k), maintenance (m), and quality assurance (n). Each department is represented by one intelligent agent. In total, we have five interacting agents as well as a special agent that rules the interaction among these agents. The description of our business model is as follows: the agent i phones the agent j asking for developing a software program. Such a request creates the strong commitment $C_1 = SCC(i, j, t, r)$, which means that the customer service i commits to the customer j to gather program requirements r if i sends a formal request t describing the required information about her program. When i receives the formal request, she delegates C_1 to the information technology agent k (i.e., $DeS(i, k, SCC(i, j, t, r))$ to gather the technical information on her behalf. The agent k fulfills the commitment C_1 by gathering all the customer requirements. By doing so, the customer j strongly commits to customer service i to send the confirmation q if she accepts the prototype of the program p: $C_2 = SCC(j, i, p, q)$. Therefore, the customer service updates both the information technology department by assigning the commitment C_2 to the agent k (i.e., $AsS(i, k, SCC(j, i, p, q))$) and the maintenance department by sending a request message asking for contacting the quality assurance department. This request

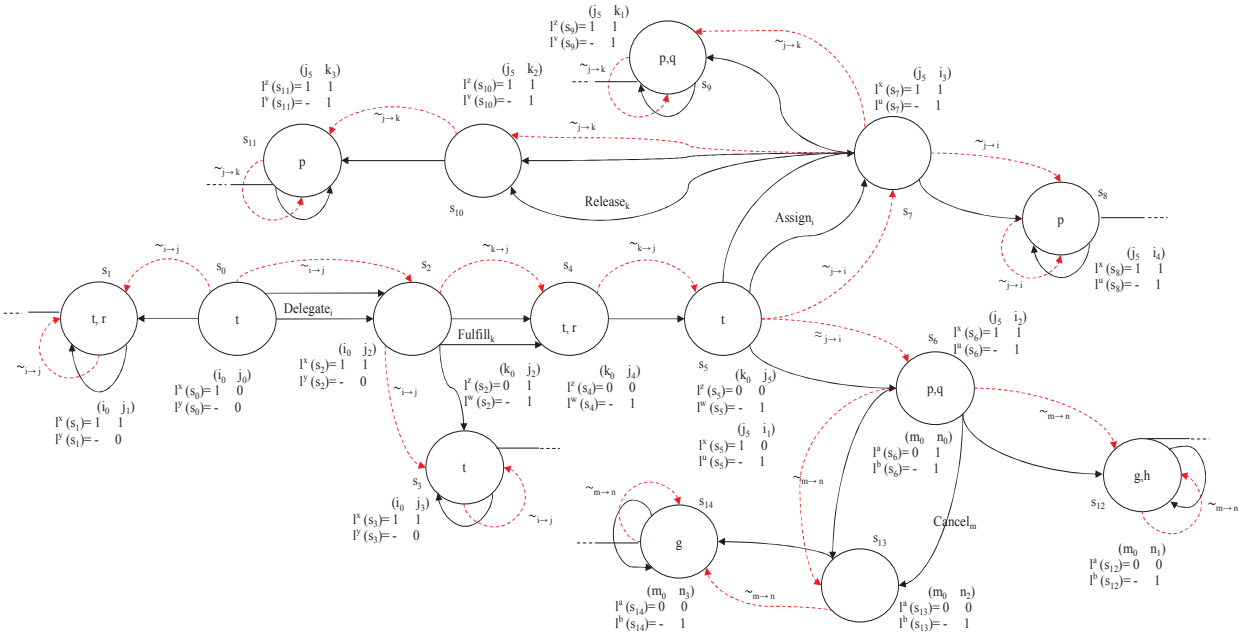

Figure 12: Our running business model

results in creating a new strong commitment (i.e., $C_3 = SCC(m, n, g, h)$) from the maintenance agent m to the quality assurance agent n to send the maintenance report h when the program is successfully deployed g. However, the information technology releases the commitment C_2, because the customer adds more requirements along with her conformation report. The release action leads to the maintenance agent to cancel her commitment C_3, which is no longer needed.

4.2.3 Algorithms of conditional commitments

The strong conditional commitment algorithm $SMC_{scc}(i, j, \psi, \varphi, M)$ reported in Algorithm 5 works as follows. It begins by computing the sets X and Y and then proceeds to

Algorithm 5 $SMC_{scc}(i, j, \psi, \varphi, M)$: the set $[\![SCC(i, j, \psi, \varphi)]\!]$

1: $X \leftarrow SMC(\psi, M)$;
2: $Y \leftarrow SMC(\neg\varphi, M)$;
3: $Z \leftarrow \{s \in S \mid \exists s' \in X \text{ s.t. } s \in SMC_{\sim}(i, j, s', M)\}$;
4: $W \leftarrow \{s \in S \mid \exists s' \in X \cap Y \text{ s.t. } s \in SMC_{\sim}(i, j, s', M)\}$;
5: return $Z - W$;

build the set Z (respectively, W) of states that can see by means of $\sim_{i\to j}$ a state s' satisfying ψ (respectively, ψ and $\neg\varphi$). Finally, it returns $Z-W$, which in turn is the set of states having accessible states satisfying the antecedent and consequence.

Example 4.2.1. *With respect to the model in Figure 12, the computation of Algorithm 5 corresponding to the commitment* $C_1 = SCC(i, j, t, r)$ *is as follows:* $X = \{s_0, s_1, s_3, , s_4, s_5\}$, $Y = \{s_0, s_2, s_3, s_5, s_6, s_7, s_8, s_9, s_{10}, s_{11}, s_{12}, s_{13}, s_{14}\}$, $Z = \{s_0, s_1, s_2, s_3\}$, *and* $W = \{s_2, s_3\}$. *Finally, the algorithm returns* $Z - W = \{s_0, s_1\}$.

Algorithm 6 reports the procedure for the weak conditional commitment modality, which in turn returns the set of states satisfying $WCC(i, j, \psi, \varphi)$, i.e., $[\![WCC(i, j, \psi, \varphi)]\!]$. First, the algorithm computes the sets X and Y of states satisfying ψ and $\neg\varphi$ respectively; then constructs the set Z of all the states that can "see" by means of $\sim_{i\to j}$ a state s' satisfying ψ and $\neg\varphi$ (i.e., s' in $X \cap Y$). Finally, the procedure returns the complement of the set Z. Suppose $\varphi = WCC(i, j, WCC(i, j, \varphi_1, \varphi_2), \varphi_3))$. By checking the structure of the

Algorithm 6 $SMC_{wcc}(i, j, \psi, \varphi, M)$: the set $[\![WCC(i, j, \psi, \varphi)]\!]$

1: $X \leftarrow SMC(\psi, M)$;
2: $Y \leftarrow SMC(\neg\varphi, M)$;
3: $Z \leftarrow \{s \in S \mid \exists s' \in X \cap Y \text{ s.t. } s \in SMC_{\sim}(i, j, s', M)\}$;
4: return $S - Z$;

formula φ, the main algorithm $SMC(\varphi, M)$ will call $SMC_{wcc}(i, j, \psi, \varphi_3, M)$ where $\psi = WCC(i, j, \varphi_1, \varphi_2)$. According to the first step in Algorithm 6, the main algorithm is called with $\psi = WCC(i, j, \varphi_1, \varphi_2)$, which in turn calls $SMC_{wcc}(i, j, \varphi_1, \varphi_2, M)$. When the set of states satisfying $WCC(i, j, \varphi_1, \varphi_2)$ (i.e., $S - Z$) is computed, then this set is passed to $SMC_{wcc}(i, j, \psi, \varphi_3, M)$, which is specifically stored in X; so the algorithm proceeds to compute the required sets Y, and Z, and then returns the set $[\![WCC(i, j, WCC(i, j, \varphi_1, \varphi_2), \varphi_3)]\!]$.

4.2.4 Algorithms of fulfilling conditional commitments

The algorithm $SMC_{fus}(i, j, \psi, \varphi, M)$ (cf. Algorithm 7) starts by calculating the set X of states satisfying the strong commitment $SCC(i, j, \psi, \varphi)$. It then constructs the set Y of states that satisfies ψ and $\neg SCC(i, j, \psi, \varphi)$. The algorithm then returns the states in Y that can be seen and reached from a state in X by means of $\sim_{i\to j}$ and local labeled transition with fulfill action computed by R_{c_i}.

Algorithm 7 $SMC_{fus}(i, j, \psi, \varphi, M)$: the set $[\![FuS(i, SCC(i, j, \psi, \varphi))]\!]$

1: $X \leftarrow SMC_{scc}(i, j, \psi, \varphi, M)$;
2: $Y \leftarrow SMC(\psi, M) \cap (S - X)$;
3: $Z \leftarrow \{s \in Y \mid \exists s' \in X \cap SMC_{\sim}(i, j, s, M) \cap SMC_{R_{ci}}(i, j, Fulfill_i, s, M)\}$;
4: return Z;

Example 4.2.2. *From Figure 12, the computation of Algorithm 7 regarding the fulfillment of the strong commitment* $SCC(k, j, t, r)$ *(i.e.,* $FuS(k, SCC(k, j, t, r))$ *is as follows:* $X = \{s_2\}$, $Y = \{s_0, s_1, s_3, s_4, s_5\}$, *and* $Z = \{s_4\}$. *The algorithm then returns* s_4.

Algorithm 8 for the fulfilment of weak commitments is very similar to Algorithm 7, except line 2, which computes the set of states satisfying φ instead of ψ in Algorithm 7.

Algorithm 8 $SMC_{fuw}(i, j, \psi, \varphi, M)$: the set $[\![FuW(i, SCC(i, j, \psi, \varphi))]\!]$

1: $X \leftarrow SMC_{wcc}(i, j, \psi, \varphi, M)$;
2: $Y \leftarrow SMC(\varphi, M) \cap (S - X)$;
3: $Z \leftarrow \{s \in Y \mid \exists s' \in X \cap SMC_{\sim}(i, j, s, M) \cap SMC_{R_{ci}}(i, j, Fulfill_i, s, M)\}$;
4: return Z;

4.2.5 Algorithms of canceling conditional commitments

The algorithm of canceling strong commitment $SMC_{cas}(i, j, \psi, \varphi, M)$ (respectively, weak commitment $SMC_{caw}(i, j, \psi, \varphi, M)$) reported in Algorithm 9 (respectively, Algorithm 10) begins with calculating the set X of states satisfying the strong commitment $SCC(i, j, \psi, \varphi)$ (respectively, weak commitment $WCC(i, j, \psi, \varphi)$) and in turn builds the set Y of states that satisfies both $\neg\psi$ and $\neg SCC(i, j, \psi, \varphi)$ (respectively, $\neg\varphi$ and $\neg WCC(i, j, \psi, \varphi)$). By do-

Algorithm 9 $SMC_{cas}(i, j, \psi, \varphi, M)$: the set $[\![CaS(i, SCC(i, j, \psi, \varphi))]\!]$

1: $X \leftarrow SMC_{scc}(i, j, \psi, \varphi, M)$;
2: $Y \leftarrow SMC(\neg\psi, M) \cap (S - X)$;
3: $Z \leftarrow \{s \in Y \mid \exists s' \in X \cap SMC_{\sim}(i, j, s, M) \cap SMC_{R_{ci}}(i, j, Cancel_i, s, M)\}$;
4: return Z;

ing so, the algorithm returns the states in Y that can be seen and reached from a state in X by means of $\sim_{i \to j}$ and local labeled transition with cancel action computed by R_{c_i}.

Example 4.2.3. *From Figure 12, the computation of Algorithm 9 regarding the cancelation of the strong commitment* $SCC(m, n, g, h)$ *(i.e.,* $CaS(m, SCC(m, n, g, h))$ *is as follows:*

Algorithm 10 $SMC_{caw}(i, j, \psi, \varphi, M)$: the set $[\![CaW(i, WCC(i, j, \psi, \varphi))]\!]$

1: $X \leftarrow SMC_{wcc}(i, j, \psi, \varphi, M)$;
2: $Y \leftarrow SMC(\neg\varphi, M) \cap (S - X)$;
3: $Z \leftarrow \{s \in Y \mid \exists s' \in X \cap SMC_{\sim}(i, j, s, M) \cap SMC_{R_{ci}}(i, j, Cancel_i, s, M)\}$;
4: return Z;

$X = \{s_6, s_{12}\}$, *and* $Y = \{s_0, s_1, s_2, s_3, s_4, s_5, s_7, s_8, s_9, s_{10}, s_{11}, s_{13}\}$ *and the algorithm returns* $Z = \{s_{13}\}$.

4.2.6 Algorithms of releasing conditional commitments

The algorithm of releasing strong commitment $SMC_{res}(i, j, \psi, \varphi, M)$ (respectively, weak commitment SMC_{rew} (i, j, ψ, φ, M)) reported in Algorithm 11 (respectively, Algorithm 12) begins with calculating the set X of states satisfying the strong commitment $SCC(i, j, \psi, \varphi)$ (respectively, weak commitment $WCC(i, j, \psi, \varphi)$) and in turn builds the set Y of states that satisfies both $\neg\psi$ and $\neg SCC(i, j, \psi, \varphi)$ (respectively, $\neg\varphi$ and $\neg WCC(i, j, \psi, \varphi)$). By

Algorithm 11 $SMC_{res}(i, j, \psi, \varphi, M)$: the set $[\![ReS(j, SCC(i, j, \psi, \varphi))]\!]$

1: $X \leftarrow SMC_{scc}(i, j, \psi, \varphi, M)$;
2: $Y \leftarrow SMC(\neg\psi, M) \cap (S - X)$;
3: $Z \leftarrow \{s \in Y \mid \exists s' \in X \cap SMC_{\sim}(i, j, s, M) \cap SMC_{R_{cj}}(i, j, Release_j, s, M)\}$;
4: return Z;

doing so, the algorithm returns the states in Y that can be seen and reached from a state in X by means of $\sim_{i\to j}$ and local labeled transition with release action computed by R_{c_j}.

Algorithm 12 $SMC_{rew}(i, j, \psi, \varphi, M)$: the set $[\![ReW(j, WCC(i, j, \psi, \varphi))]\!]$

1: $X \leftarrow SMC_{wcc}(i, j, \psi, \varphi, M)$;
2: $Y \leftarrow SMC(\neg\varphi, M) \cap (S - X)$;
3: $Z \leftarrow \{s \in Y \mid \exists s' \in X \cap SMC_{\sim}(i, j, s, M) \cap SMC_{R_{cj}}(i, j, Release_j, s, M)\}$;
4: return Z;

Example 4.2.4. *From Figure 12, the computation of Algorithm 11 with regard to release the strong commitment* $SCC(m, n, g, h)$ *(i.e.,* $ReS(k, SCC(j, k, p, q))$*) is as follows:* $X = \{s_7, s_9\}$, *and* $Y = \{s_0, s_1, s_2, s_3, s_4, s_5, s_{10}, s_{12}, s_{13}, s_{14}\}$ *and then the algorithm returns* $Z = \{s_{10}\}$.

4.2.7 Algorithms of delegating conditional commitments

The algorithm of delegating strong commitment $SMC_{des}(i, j, k, \psi, \varphi, M)$ (respectively, weak commitment $SMC_{dew}(i, j, k, \psi, \varphi, M)$) reported in Algorithm 13 (respectively, Algorithm 14) begins with calculating the set W of states satisfying the strong commitment $SCC(i, j, \psi, \varphi)$ (respectively, weak commitment $WCC(i, j, \psi, \varphi)$) and in turn builds the set X of states that satisfies the strong commitment $SCC(k, j, \psi, \varphi)$ (respectively, weak commitment $WCC(k, j, \psi, \varphi)$). Afterwards, it computes the set Y of states in X that sat-

Algorithm 13 $SMC_{des}(i, j, k, \psi, \varphi, M)$: the set $[\![DeS(i, k, SCC(i, j, \psi, \varphi))]\!]$

1: $W \leftarrow SMC_{scc}(i, j, \psi, \varphi, M)$;
2: $X \leftarrow SMC_{scc}(k, j, \psi, \varphi, M)$;
3: $Y \leftarrow SMC(\neg\psi, M) \cap (S - W) \cap X$;
4: $Z \leftarrow \{s \in Y \mid \exists s' \in W \cap SMC_{\sim}(i, j, s, M) \cap SMC_{R_{ci}}(i, j, Delegate_i, s, M)\}$;
5: return Z;

isfies $\neg\psi$ and $\neg SCC(i, j, \psi, \varphi)$ (respectively, $\neg\varphi$ and $\neg WCC(i, j, \psi, \varphi)$). The algorithm finally returns the states in Y that can be seen and reached from a state in W by means of $\sim_{i\to j}$ and local labeled transition with delegate action computed by R_{c_i}.

Algorithm 14 $SMC_{dew}(i, j, k, \psi, \varphi, M)$: the set $[\![DeW(i, k, WCC(i, j, \psi, \varphi))]\!]$

1: $W \leftarrow SMC_{wcc}(i, j, \psi, \varphi, M)$;
2: $X \leftarrow SMC_{wcc}(k, j, \psi, \varphi, M)$;
3: $Y \leftarrow SMC(\neg\varphi, M) \cap (S - W) \cap X$;
4: $Z \leftarrow \{s \in Y \mid \exists s' \in W \cap SMC_{\sim}(i, j, s, M) \cap SMC_{R_{ci}}(i, j, Delegate_i, s, M)\}$;
5: return Z;

Example 4.2.5. *From Figure 12, the computation of Algorithm 13 regarding the delegation of the strong commitment $SCC(i, j, t, r)$ (i.e., $DeS(i, k, SCC(i, j, t, r))$ is as: $W = \{s_0, s_1\}$, $X = \{s_2\}$, $Y = \{s_2\}$ and then the algorithm returns $Z = \{s_2\}$.*

4.2.8 Algorithms of assigning conditional commitments

The algorithm of assigning strong commitment $SMC_{ass}(i, j, k, \psi, \varphi, M)$ (respectively, weak commitment $SMC_{asw}(i, j, k, \psi, \varphi, M)$) reported in Algorithm 15 (respectively, Algorithm 16) begins with calculating the set W of states satisfying the strong commitment $SCC(i, j, \psi, \varphi)$ (respectively, weak commitment $WCC(i, j, \psi, \varphi)$) and in turn builds the set X of states that satisfies the strong commitment $SCC(i, k, \psi, \varphi)$ (respectively, weak commitment $WCC(i, k, \psi, \varphi)$). Afterwards, it computes the set Y of states in X that satisfies

Algorithm 15 $SMC_{ass}(i, j, k, \psi, \varphi, M)$: the set $[\![AsS(j, k, SCC(i, j, \psi, \varphi))]\!]$

1: $W \leftarrow SMC_{scc}(i, j, \psi, \varphi, M)$;
2: $X \leftarrow SMC_{scc}(i, k, \psi, \varphi, M)$;
3: $Y \leftarrow SMC(\neg\psi, M) \cap (S - W) \cap X$;
4: $Z \leftarrow \{s \in Y \mid \exists s' \in W \cap SMC_{\sim}(i, j, s, M) \cap SMC_{R_{cj}}(i, j, Assign_j, s, M)\}$;
5: return Z;

$\neg\psi$ and $\neg SCC(i, j, \psi, \varphi)$ (respectively, $\neg\varphi$ and $\neg WCC(i, j, \psi, \varphi)$). The algorithm finally returns the states in Y that can be seen and reached from a state in W by means of $\sim_{i\to j}$ and local labeled transition with assign action computed by R_{c_j}.

Algorithm 16 $SMC_{asw}(i, j, k, \psi, \varphi, M)$: the set $[\![AsW(j, k, WCC(i, j, \psi, \varphi))]\!]$

1: $W \leftarrow SMC_{wcc}(i, j, \psi, \varphi, M)$;
2: $X \leftarrow SMC_{wcc}(i, k, \psi, \varphi, M)$;
3: $Y \leftarrow SMC(\neg\varphi, M) \cap (S - W) \cap X$;
4: $Z \leftarrow \{s \in Y \mid \exists s' \in W \cap SMC_{\sim}(i, j, s, M) \cap SMC_{R_{cj}}(i, j, Assign_j, s, M)\}$;
5: return Z;

Example 4.2.6. *From Figure 12, the computation of Algorithm 15 regarding the assignment of the strong commitment* $SCC(j, i, p, q)$ *(i.e.,* $AsS(i, k, SCC(j, i, p, q))$*) is as:* $W = \{s_5\}$, $X = \{s_7, s_9\}$, $Y = \{s_7\}$ *and then the algorithm returns* $Z = \{s_7\}$.

4.3 Theoretical results

In this section, we investigate important theoretical results from the algorithmic point of view.

4.3.1 Termination

Proposition 4.3.1. *The developed procedures reported in Algorithms 5 to 16 terminate.*

Proof. Since the set of reachable states[3] employed in such procedures can be easily computed by a monotonic operator that has a fixed point, then the developed procedures terminate[4]. Notice that since the set of social states is finite and the operator is monotonic, it will admit a least fixed point that can be calculated by iterating over empty set. □

[3]This set is defined by the states reachable from the set of initial states using temporal transitions.

[4]The similar argument is applied to prove the termination of the procedures employed for epistemic operators [64] that have the same structure as ours.

The following proposition is a direct consequence of Proposition 4.3.1.

Proposition 4.3.2 (Termination)**.** *Let* $M = (S, I, T, \{\sim_{i\to j} \mid (i,j) \in \mathcal{A}^2\}, \{R_{c_i} \mid i \in \mathcal{A}\}, \mathcal{V})$ *be a finite model for CTL*$^{cc,\alpha}$*. Assume the CTL procedures are terminating. Then, the symbolic model checking algorithm* SMC *as defined in Algorithm 2 also terminates.*

4.3.2 Soundness

The correctness of the developed model checking algorithm reported in Algorithm 2 is established by the correctness of model checking procedures developed for: 1) CTL-temporal operators, which in fact derive from the correctness of the procedures employed in the verification of standard CTL models [32]; and 2) conditional commitment operators and commitment action operators. To support the second argument, we need to show that when the developed algorithms terminate, then there is a set of states satisfying the formula in question.

Proposition 4.3.3. *Let* Y*,* Z*, and* W *be the sets of states computed using the algorithms invoked in the left side of the following equivalences:*

1. *When Algorithm 5 terminates, we have* $(M,s) \models SCC(i,j,\psi,\varphi)$ *iff* $s \in Z - W$.
2. *When Algorithm 6 terminates, we have* $(M,s) \models WCC(i,j,\psi,\varphi)$ *iff* $s \in S - Z$.
3. *When Algorithm 7 terminates, we have* $(M,s) \models FuS(i, SCC(i,j,\psi,\varphi))$ *iff* $s \in Z$.
4. *When Algorithm 8 terminates, we have* $(M,s) \models FuW(i, WCC(i,j,\psi,\varphi))$ *iff* $s \in Z$.
5. *When Algorithm 9 terminates, we have* $(M,s) \models CaS(i, SCC(i,j,\psi,\varphi))$ *iff* $s \in Z$.
6. *When Algorithm 10 terminates, we have* $(M,s) \models CaW(i, WCC(i,j,\psi,\varphi))$ *iff* $s \in Z$.
7. *When Algorithm 11 terminates, we have* $(M,s) \models ReS(j, SCC(i,j,\psi,\varphi))$ *iff* $s \in Z$.
8. *When Algorithm 12 terminates, we have* $(M,s) \models ReW(j, WCC(i,j,\psi,\varphi))$ *iff* $s \in Z$.
9. *When Algorithm 13 terminates, we have* $(M,s) \models DeS(i,k, SCC(i,j,\psi,\varphi))$ *iff* $s \in Z$.
10. *When Algorithm 14 terminates, we have* $(M,s) \models DeW(i,k, WCC(i,j,\psi,\varphi))$ *iff* $s \in Z$.

11. *When Algorithm 15 terminates, we have* $(M, s) \models AsS(j, k, SCC(i, j, \psi, \varphi))$ *iff* $s \in Z$.

12. *When Algorithm 16 terminates, we have* $(M, s) \models AsW(j, k, WCC(i, j, \psi, \varphi))$ *iff* $s \in Z$.

Proof. Since Algorithm 6 comes directly from the defined semantics, we show the proof by induction over the structure of the algorithm. Specifically, lines 1–2 of the algorithm correspond to the base case: the two sets X and Y compute the states satisfying the antecedent ψ and the negation of the consequence φ, respectively. For the efficiently purpose, we compute $\neg\varphi$ instead of φ so as to use the existential operator $\exists$ to compute the members of the set Z instead of $\forall$. When the computation of X (or Y) produces an empty set, the algorithm returns in line 4 the set S, i.e., all model's states.

Line 3 corresponds to the induction hypothesis step, i.e., there is at least a state satisfying ψ and $\neg\varphi$; so $X \cap Y$ is not empty. Given that, the algorithm proceeds to compute the set Z of preimage states of this state by calling Algorithm 3. Line 4 corresponds to the induction step: by removing from the world S the states in Z that can see via $\sim_{i\to j}$ a state satisfying ψ and $\neg\varphi$ ($S - Z$), we deduce that the rest of states in S should satisfy $WCC(i, j, \psi, \varphi)$. Other parts of the proposition can be proved by a similar way. □

Having proved the correctness of the developed algorithms, the following proposition is a direct result.

Proposition 4.3.4 (Correctness)**.** *The developed symbolic model checking algorithm reported in Algorithm 2 and symbolic model checker reported in Algorithm 1 are correct.*

4.4 Implementation

Rather than implementing our symbolic model checker from scratch, we have adopted and extended the symbolic model checker MCMAS [64]. MCMAS is the only symbolic model checker, which supports the semantics of interpreted systems (i.e., MAS models) and specifications based on CTL. It also performs OBDD operations by means of the efficient CUDD library. A distribution of MCMAS is publicly available, which is equipped with the source code and case studies. Moreover, it has been successfully adopted to model check web service composition [65], multi-agent interaction protocols [49, 53, 52], community-based agent web services [16], and industrial models [15]. Our extension of MCMAS tool generates an extended version of both MCMAS and its input language ISPL, which we

called respectively MCMAS+ and ISPL+. Technically, the extension process is performed in the utilities of MCMAS using the following two steps:

1. Adding 14 OBDD symbolic algorithms, which encode and implement our algorithms developed and dedicated to CTL$^{cc,\alpha}$ into the corresponding directory of the MCMAS tool (these algorithms were implemented in C++).
2. Augmenting the CTL$^{cc,\alpha}$ syntax on top of the standard CTL's parser, syntax checker (using the method "check_formulae" in the "modal_formulae" class), deriver and header types.

To achieve step (1), the following two steps are performed:

– All components of the model M as well as the states and sets of states computed by the developed algorithms are represented as Boolean formulae in which all operations on sets are transformed into operations on Boolean formulae. For instance, the intersection of two sets transforms into the multiplication of two Boolean formulae encoding such two sets. The resulting Boolean formulae can be readily encoded as OBDDs (refer to [32] for more details). The following is our OBDD implementation of the strong delegation algorithm (see Algorithm 13):

```
BDD get_DES_states_new(BDD p, BDD q, string name1, string
name2, string name3,  bdd_parameters *para)
  {
  vector<basictype *> shared;
  vector<basictype *> shared1;
  get_shared_vars(name1, name2, &shared);
  get_shared_vars(name3, name2, &shared1);
  BDD W = get_SCC_states(p, q, name1, name2, para);
  BDD X = get_SCC_states(p, q, name3, name2, para);
  BDD Y = (*para->reach - p) * (*para->reach - W) * X;
  BDD Z = build_accessible_equivalence(name1, name2,
  &shared, para);
  string action_name = "Delegate_" + name1;
  basic_agent *agent = (*is_agents)[name1];
  BDD a = agent->encode_action(para, action_name);
  BDD result = Y * get_accessible_image_new(W, Z, a, para);
```

```
return result;
}
```

– Two OBDD algorithms are added into the corresponding directory of the MCMAS tool in order to respectively compute the set of shared variables for any pair (i, j) of agents and build an equivalence relation R such that for any states s_1 and s_2, $(s_1, s_2) \in R$ iff the shared variables have the same values in the i and j's local states. Our extension currently supports all MCMAS data types (e.g., enumeration, bounded integer, and Boolean) to define shared and unshared variables. These algorithms were also implemented in C++. Other algorithms and methods are added to make our implementation compatible with the modularity feature. This extension doesn't affect the performance of the existing CTL temporal operators. Finally, our extension was released as an independent open-source tool under GPL.

On a more down-to-earth note, MCMAS features a graphical user interface (based on Eclipse) that supports a wide range of features. These features can guide the users and designers to create, edit and track modeled systems by carrying out dynamic syntax checking along with an additional ANTLR parser. Its interface also supports displaying counter and witness examples, outline view, text marking and formatting, syntax highlighting, and content assist automatically. Moreover, thanks to its embedding in a Java archive, such a graphical user interface could be seamlessly integrated with other applications that need model checking conditional commitments or other properties of the domain. For better usability and ease-of-use the MCMAS's graphical user interface, we:

1. Extended the implementation of the specialized system description language ISPL to include shared and unshared variables and to support the commitment modalities and the commitment action modalities (such implementations were performed in Java).
2. Developed and implemented a new facility to display labeled transition models under verification process. This facility along with the counter-example facility reduce inefficient and labor-intensive processes performed by the prospective users and designers to step and trace through the models to find the places of errors or bugs reported in counter-examples in order to fix them. All these graphs are represented by Graphviz.

In the following, we present the automated ISPL+ model, which reflects the structure of the extended interpreted systems generated by MCMAS+. For readability, we commented different sections using "- -".

```
Agent Environment --Beginning of an Environment agent section
    Vars:
    end Vars
    Actions = {};
    Protocol:
    end Protocol
    Evolution:
    end Evolution
end Agent             --Ending of an Environment agent section
Agent                 --Beginning of an agent section
    Vars:             --local variables including local states
    end Vars          --and shared and unshared variables
    Actions = {};     --local actions
    Protocol:         --local protocol
    end Protocol
    Evolution:        --local transition function
    end Evolution
end Agent
Evaluation            --evaluation function to define atomic
end Evaluation        --propositions
InitStates            --initial states
end InitStates
Formulae              --formulae being checked
end Formulae
```

We used our automated ISPL+ template generated by MCMAS+'s graphical user interface plug-in under Eclipse to encode the running business model illustrated in Figure 12. An environment agent is added to be accessible by all participating agents. This model has 6 agents, 5 conditional commitments and their negations needed to satisfy the semantics of commitment actions as well as 6 conditional commitment actions. Figure 13 shows the labeled transition system of this model automatically generated by our tool. Furthermore, we expressed the following reachability properties using CTL$^{cc,\alpha}$ to verify the reliability of the model:

1. $EF\ DeS(i, k, SCC(i, j, t, r))$
2. $EF\ FuS(k, SCC(k, j, t, r))$

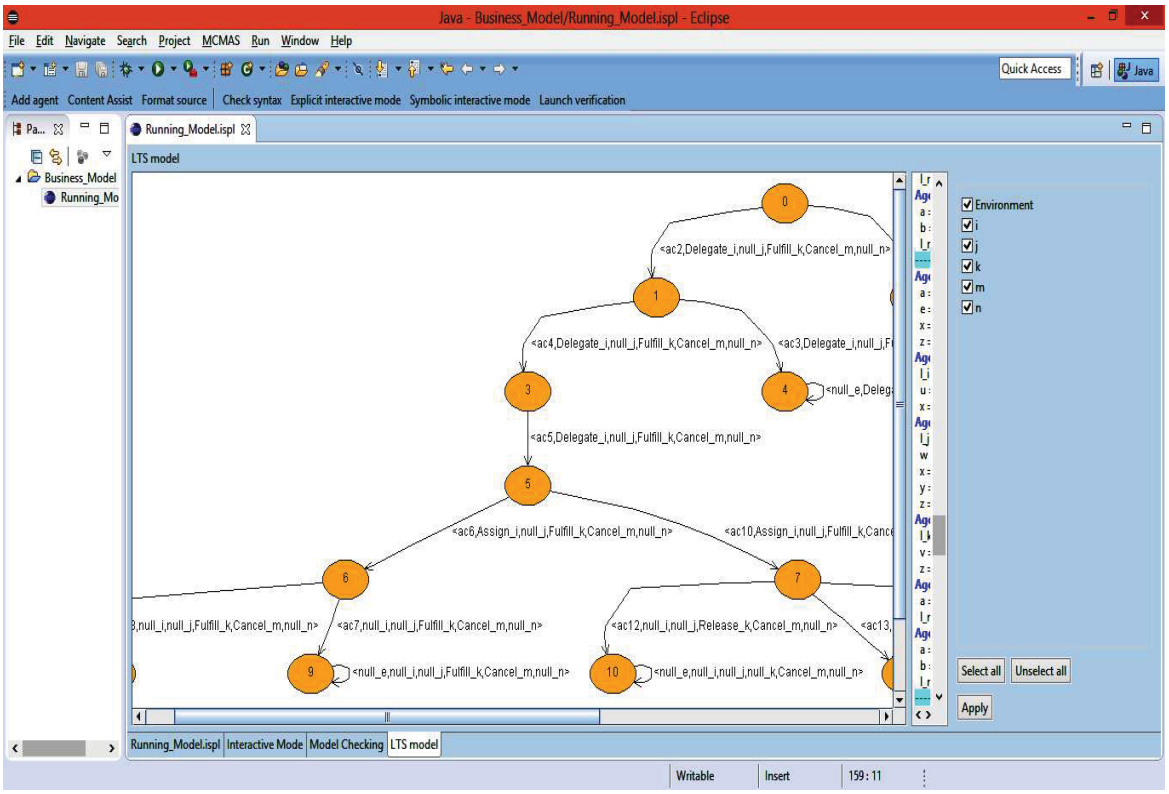

Figure 13: The generated labeled transition system of the running model

3. $EF\ FuW(k, SCC(k, j, t, r))$
4. $EF\ CaS(m, SCC(m, n, g, h))$
5. $EF\ AsS(i, k, SCC(j, i, p, q))$
6. $EF\ ReS(k, SCC(j, k, p, q))$

The first formula means that there is a path and in its future the agent i can delegate her strong commitment to the agent k. Other formulae can be read in the same way. Although these formulae seem simple, they are reasonable and enough to test: 1) the implementation of our algorithms; and 2) the existence of commitment actions, which constitute all business scenarios shaping the model. Moreover, such action-based properties cannot be expressed and directly model checked in any existing approach discussed in Chapter 2. When the verification results of these properties return true, then the commitments invoked in them hold as well (see the semantics of commitment actions in Chapter 3). For example, to fulfill a commitment, the commitment should be first activated. This is why we did not check first the existence of commitments. Figure 14 shows the verification results of our

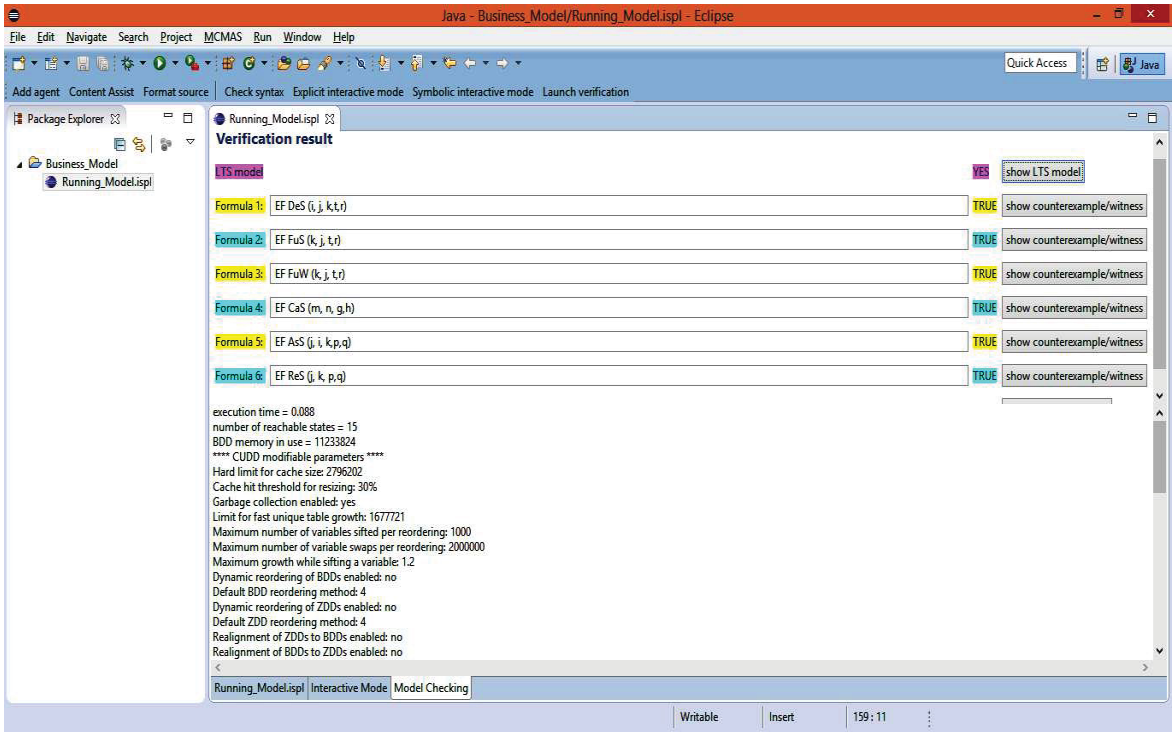

Figure 14: Verification results of our running model

tested properties. In the figure, all these properties are evaluated to true. It is important to mention that the graphical user interface reports some statistical results such as the execution time (0.088 second), the number of reachable states (15 states), and the memory in use (11 Megabytes). To evaluate the usability of the extended graphical user interface, we conducted an empirical study involving 6 graduate students in the lab having a prior experience in model-checking: 2 had more than four years; 2 had one to three years; and 2 had one year. The study divided the students into two groups (Group I and Group II), who used two versions of the graphical user interface to model the same business scenario. The students had fifteen days time to complete the model encoding process. We provided a description with the same level of details of each graphical user interface and business model to the students. The students in Group I modeled the above business model using the old graphical user interface and students in Group II did the same using the new graphical user interface. Each student worked individually. We found that Group II finished the modeling process easier and faster than Group I and when we reviewed their modelings, we found that they are very close to our modeling. Moreover, one member of Group I did not complete the modeling process in the predefined time.

Chapter 5

Computational Complexity Analysis

This chapter[1] covers:

- The time complexity of the model checking problem of CTL$^{cc,\alpha}$, which addresses the first part of Question 1.3.7 mentioned in Chapter 1.
- The space complexity of the model checking problem of CTL$^{cc,\alpha}$, which addresses the second part of Question 1.3.7 mentioned in Chapter 1.

5.1 Introduction

As we shown in [52, 39], the motivations of studying the computational complexity of model checking algorithm are to: 1) find a formal argument that shows the performance of the proposal; 2) compute resources sufficient for tackling all problem's instances including the worst case [77]; 3) give a clear picture of the actual computational difficulty behind the problem; and 4) compare different model checking techniques. Figure 15 shows the hierarchical relationship between the common complexity classes. These classes can be read from bottom to up as *logarithmic space, nondeterministic logarithmic space, polynomial time, nondeterministic polynomial time, polynomial space, nondeterministic polynomial space*, and *exponential time*.

5.2 Time complexity

In this section, we will prove that the CTL$^{cc,\alpha}$ model checking algorithm is P-complete. This result means that our algorithm can be run in a polynomial time with respect to the

[1] The results in this chapter are collected from our publications in [39, 44, 52].

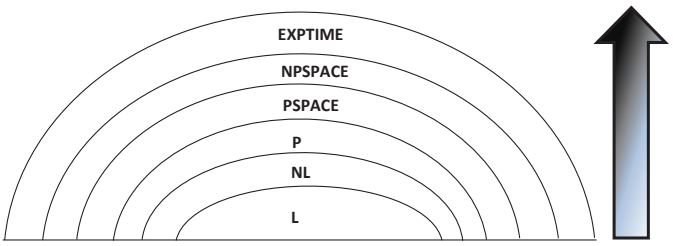

Figure 15: The relationship between common complexity classes

size of the model and the length of the formula.

Theorem 5.2.1. *Let $|M|$ and $|\varphi|$ be the size of the explicit model and length of the formula. The problem of CTL$^{cc,\alpha}$ model checking can be solved in time $\mathcal{O}(|M| \times |\varphi|)$.*

Proof. The main idea of our proof is based on the fact that CTL$^{cc,\alpha}$ is an extension of CTL with a set of temporal modalities. The CTL model checking for explicit models is linear in the size of the model and length of the formula, the proof is introduced in [32]. Therefore, we solely need to study the time complexity of our Algorithms from 5 to 16. Steps 2 and 3 in Algorithms from 5 to 12 and steps 3 and 4 in Algorithms from 13 to 16 are readily constructing sets of states: 1) by invoking the CTL algorithm; or 2) by performing comparison operations on states, which can be easily done in a linear running time. In case of Algorithms 3 and 4, the argument of performing the comparison operations on states is also valid. Moreover, step 1 in Algorithms from 7 to 12 and steps 1 and 2 in Algorithms from 13 to 16 recursively call the model checking algorithm on the sub-formulae ψ_1 and ψ_2 of the formula $\varphi = SCC(i, j, \psi_1, \psi_2)$ (respectively, $\varphi = WCC(i, j, \psi_1, \psi_2)$). Assume $\varphi = SCC(i, j, \psi_1, \psi_2)$, then Algorithm 5 of strong commitments is recursively invoked until a CTL sub-formula is found. The depth of the recursion is therefore limited to the length of the formula φ. Clearly, this depth is linear in the length of the formula φ. The same arguments are valid in the case of weak commitments. Since CTL model checking is linear in both the size of the model and the length of the formula, we then conclude that CTL and CTL$^{cc,\alpha}$ algorithms have the same complexity; so the theorem. □

Theorem 5.2.2. *The problem of CTL$^{cc,\alpha}$ model checking is P-complete.*

Proof. The upper bound (i.e., membership) is in P, which follows from Theorem 5.2.1. The lower bound (i.e., hardness) is in P, which follows from a log-space reduction from CTL model checking algorithm proved to be P-complete in [77]. □

5.3 Space complexity using reduction technique

Hereafter, we will prove that the space complexity of the problem of model checking CTL$^{cc,\alpha}$ is PSPACE-complete for concurrent programs. This result means that there is an algorithm solving the problem in polynomial space in the size of the components constituting concurrent programs and the length of the formula being model checked. Concurrent programs in principle provide compact representations. Analyzing the space complexity of our algorithm for concurrent programs is motivated by the fact that explicit representations are not feasible in real model checker tools such as MCMAS, SPIN, and NuSMV. These tools indeed provide compact modeling languages with a relatively high-level method to define concurrent programs.

5.3.1 Concurrent programs and proof idea

We start with formally defining concurrent programs. A concurrent program as introduced by Kupferman et al. [62] is composed of n concurrent processes (e.g., modules, protocols or agents). Each process is described by a transition system K_i defined as follows: $K_i = (AP_i, AC_i, Z_i, \Delta_i, z_i^0, H_i)$ where: AP_i is a set of local atomic propositions, AC_i is a local action alphabet, Z_i is a finite set of local states, $\Delta_i \subseteq Z_i \times AC_i \times Z_i$ is a local transition relation, $z_i^0 \in Z_i$ is an initial state, and $H_i : Z_i \to 2^{AP_i}$ is a local state labeling function. A concurrent behavior of these processes can be described using a global transition system K, which is computed by constructing the reachable states of the product of the processes and synchronization of transitions is obtained using common action names. Assume $AP = \bigcup_{i=1}^{n} AP_i$, $AC = \bigcup_{i=1}^{n} AC_i$, $S = \Pi_{i=1}^{n} Z_i$, $z^0 = (z_1^0, z_2^0, \ldots, z_n^0)$, and $H(z) = \bigcup_{i=1}^{n} H_i(s[i])$ for every $z \in Z$, where $z[i]$ is the ith component of z. Thus, $K = (AP, AC, Z, \Delta, z^0, H)$ where $(z, a, z') \in \Delta$ *iff* (1) for all $1 \leq i \leq n$ such that $a \in AC_i$, we have $(z[i], a, z'[i]) \in \Delta_i$; and (2) for all $1 \leq i \leq n$ such that $a \notin AC_i$, we have $z[i] = z'[i]$.

The idea of our proof is to employ a reduction technique to compute the lower and upper bounds of the CTL$^{cc,\alpha}$ model checking problem. This technique comprises of two incorporated steps. In the former step, we transform the model of CTL$^{cc,\alpha}$ into the model of ARCTL [73] using a log-space reduction. The latter step is responsible for transforming the CTL$^{cc,\alpha}$ formulae into the ARCTL formulae using a polynomial space reduction.

5.3.2 The ARCTL logic

The selection of ARCTL is natural and direct for two technical reasons: 1) the ARCTL model is characterized by labeled transitions with actions and during the transformation process, the commitment actions as well as other needed actions are instantly mapped into ARCTL actions; and 2) the ARCTL action formulae are used to define formulae that capture the semantic rules of the commitment action modalities and commitment modality. Particularly, ARCTL is a branching-time temporal logic, which extends CTL with action formulae. The key characteristic of ARCTL is to restrict path quantifiers with an action formula that should satisfy in order to determine sharply the precise paths to evaluate path formulae. Therefore, ARCTL is able to express state- and action-based system properties like CTL$^{cc,\alpha}$. As introduced in [73], the following BNF grammar defines the syntax of ARCTL:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid E_\alpha X\varphi \mid E_\alpha(\varphi\ U\ \varphi) \mid E_\alpha G\varphi$$
$$\alpha ::= b \mid \neg\alpha \mid \alpha \vee \alpha$$

where φ is a state formula that holds on a given state, α is an action formula, $p \in \Phi_p$ is an atomic proposition, and $b \in \Phi_b$ is an atomic action proposition.

Definition 5.3.1 (ARCTL Models)**.** *A model $M_A = (G, I_G, Act, TR, V_G, V_{Act})$ is a tuple where G is a nonempty set of states; $I_G \subseteq G$ is a set of initial states; Act is a set of actions; $TR \subseteq G \times Act \times G$ is a labeled transition relation; $V_G : G \to 2^{\Phi_P}$ is a function assigning to each state a set of atomic propositions to interpret this state; and $V_{Act} : Act \to 2^{\Phi_b}$ is a function assigning to each action a set of atomic action propositions to interpret this action.*

The semantics of this logic [73] is given by defining the α-restriction of $M_A = (G, I_G, Act, TR, V_G, V_{Act})$ as follows: $M_A^\alpha = (G, I_G, Act, TR^\alpha, V_G, V_{Act})$, where TR^α is a transition relation such that $(g, a, g') \in TR^\alpha$ iff $(g, a, g') \in TR$ and $a \models \alpha$ where $\models$ is defined as follows:

- $a \models b$ iff $b \in V_{Act}(a)$;
- $a \models \neg\alpha$ iff not $(a \models \alpha)$ and;
- $a \models \alpha \vee \alpha'$ iff $a \models \alpha$ or $a \models \alpha'$.

The intuition behind the α-restriction is to concentrate each time on certain transitions whose labels satisfy a given action formula and disregard all other transitions. This restriction is useful when checking a formula since relevant transitions are merely considered.

Specifically, the authors in [73] defined the set $\Pi^{\alpha}(g)$ of paths (called α-paths) whose actions satisfy a given action formula α and these paths start at g to formally interpret an ARCTL formula. The satisfaction relation, denoted by $(M_A^{\alpha}, g) \models \varphi$ or concisely $g \models \varphi$, is defined as follows (notice that we omit the semantics of Boolean connectives and propositional atoms):

$g \models E_{\alpha} X\varphi$ *iff* $\exists \pi \in \Pi^{\alpha}(g)$ *and* $\pi(1) \models \varphi$,

$g \models E_{\alpha}(\varphi \, U \, \psi)$ *iff* $\exists \pi \in \Pi^{\alpha}(g)$ $s.t.$ *for some* $k \geq 0, \pi(k) \models \psi$ *and* $\pi(j) \models \varphi \, \forall \, 0 \leq j \leq k-1$,

$g \models E_{\alpha} G\varphi$ *iff* $\exists \pi \in \Pi^{\alpha}(g)$ $s.t.$ $\pi(k) \models \varphi$ *for all* $k \geq 0$.

5.3.3 Transformation procedure

Our transformation procedure from the problem of model checking CTL$^{cc,\alpha}$ to the problem of model checking ARCTL is defined as follows: Given a CTL$^{cc,\alpha}$ model $M = (S, I, T, \{\sim_{i \to j} \mid (i,j) \in \mathcal{A}^2\}, \{R_{c_i} | \; i \in \mathcal{A}\}, \mathcal{V})$ and a CTL$^{cc,\alpha}$ formula φ, we have to define an ARCTL model $M_A^{\alpha} = \mathcal{F}(M)$ and an ARCTL formula $\mathcal{F}(\varphi)$ using a transformation function $\mathcal{F}$ such that $M \models \varphi$ iff $\mathcal{F}(M) \models \mathcal{F}(\varphi)$. The model $\mathcal{F}(M)$ is defined as an ARCTL model $M_A^{\alpha} = (G, I_G, Act, TR^{\alpha}, V_G, V_{Act})$ as follows:

1. $G \leftarrow S, I_G \leftarrow I, V_G \leftarrow \mathcal{V}$,
2. To define the set Act, we define: 1) the function $map : Act_i^c \to \{1,2,3,4,5\}$, which takes a set of commitment actions as input and returns a number that represents the index of the input action. It is specifically defined as: $1 = map(Fulfill)$, $2 = map(Cancel)$, $3 = map(Release)$, $4 = map(Delegate)$ and $5 = map(Assign)$; and 2) the set Φ_b from the following four sets of atomic action propositions:
   - the set $W \leftarrow \{\epsilon\}$ for the transition relation T to capture the semantics of temporal modalities.
   - the set $X \leftarrow \{\alpha_{1\to1}, \alpha_{1\to2}, \ldots, \alpha_{n\to n}\}$ for the social accessibility relation $\sim_{i\to j}$ to capture the semantics of conditional commitments.
   - the set $Y \leftarrow \{\gamma_{11}, \gamma_{12}, \ldots, \gamma_{n5}\}$ for the local transition labeled by commitment actions and defined by R_{c_i}.

– the set $Z \leftarrow \{\beta_{11\rightarrow 1}, \beta_{21\rightarrow 2}, \ldots, \beta_{5n\rightarrow n}\}$ for the symmetric closure of the social accessibility relation $\sim_{i\rightarrow j}$ and local labeled transition relation R_{c_i} to capture the semantics of commitment action modalities.

If $\Phi_b \leftarrow W \cup X \cup Y \cup Z$, then $Act \leftarrow \{\alpha^o\} \cup \{\alpha^{11}, \alpha^{12}, \ldots, \alpha^{nn}\} \cup \{\beta^{111}, \beta^{211}, \ldots, \beta^{5nn}\} \cup \{\gamma^{11}, \gamma^{12}, \ldots, \gamma^{n5}\}$ where α^o, α^{ij} and γ^{ik} are the actions labeling transitions respectively obtained from the transition relation T, social accessibility relation $\sim_{i\rightarrow j}$ and local labeled transition relation R_{c_i}. β^{kij} is the action labeling the symmetric transitions added when there exists transitions labeled with α^{ij} and γ^{ik} and needed to define transformation of the action formulae,

3. The function V_{Act} is defined as follows:
   – If $\alpha^o \in Act$, then $V_{Act}(\alpha^o) \leftarrow \{\epsilon\}$,
   – If $\alpha^{ij} \in Act$, then $V_{Act}(\alpha^{ij}) \leftarrow \{\alpha_{i\rightarrow j}\}$ for $1 \leq i \leq n$ and $1 \leq j \leq n$,
   – If $\beta^{kij} \in Act$, then $V_{Act}(\beta^{kij}) \leftarrow \{\beta_{ki\rightarrow j}\}$ for $1 \leq k \leq 5$, $1 \leq i \leq n$ and $1 \leq j \leq n$,
   – If $\gamma^{ik} \in Act$, then $V_{Act}(\gamma^{ik}) \leftarrow \{\gamma_{ik}\}$ for $1 \leq i \leq n$ and $1 \leq k \leq 5$, and
4. The labeled transition relation TR^{α} combines the temporal transition T, accessibility relation $\sim_{i\rightarrow j}$ and local labeled transition relation R_{c_i} and their symmetric closures under the following conditions: for states $s, s' \in S$,
   – If $(s, s') \in T$, then $(g, \alpha^o, g') \in TR^{\epsilon}$,
   – If $s \sim_{i\rightarrow j} s'$, then $(g, \alpha^{ij}, g') \in TR^{\alpha_{i\rightarrow j}}$,
   – If $(l_i(s), a_i, l_i(s')) \in R_{c_i}$ and $a \in Act_i^c$, then $(g, \gamma^{ik}, g') \in TR^{\gamma_{ik}}$ where $k = map(a)$.
   – If $(g, \alpha^{ij}, g') \in TR^{\alpha_{i\rightarrow j}}$ and $(g, \gamma^{ik}, g') \in TR^{\gamma_{ik}}$, then $(g', \beta^{kij}, g) \in TR^{\beta_{ki\rightarrow j}}$.

Let us now define $\stackrel{\mathcal{F}}{=}$ as equal under the transformation function $\mathcal{F}$ and $\mathcal{F}(\varphi)$ as an ARCTL formula by induction on the form of the CTL$^{cc,\alpha}$ formula φ.

1. $\mathcal{F}(p) \stackrel{\mathcal{F}}{=} p$, if p is an atomic proposition;
2. $\mathcal{F}(\neg\varphi) \stackrel{\mathcal{F}}{=} \neg\mathcal{F}(\varphi)$; $\mathcal{F}(\varphi \vee \psi) \stackrel{\mathcal{F}}{=} \mathcal{F}(\varphi) \vee \mathcal{F}(\psi)$; $\mathcal{F}(EX\varphi) \stackrel{\mathcal{F}}{=} EX\mathcal{F}(\varphi)$;
3. $\mathcal{F}(E(\varphi\, U\, \psi)) \stackrel{\mathcal{F}}{=} E(\mathcal{F}(\varphi)\, U\, \mathcal{F}(\psi))$;
4. $\mathcal{F}(EG\varphi) \stackrel{\mathcal{F}}{=} EG\mathcal{F}(\varphi)$;

5. $\mathcal{F}(WCC(i,j,\psi,\varphi)) \stackrel{\mathcal{F}}{=} A_{\alpha^{ij}} X(\mathcal{F}(\psi) \rightarrow \mathcal{F}(\varphi))$;
6. $\mathcal{F}(SCC(i,j,\psi,\varphi)) \stackrel{\mathcal{F}}{=} E_{\alpha^{ij}} X\mathcal{F}(\psi) \wedge \mathcal{F}(WCC(i,j,\psi,\varphi))$;
7. $\mathcal{F}(FuS(i,SCC(i,j,\psi,\varphi))) \stackrel{\mathcal{F}}{=} E_{\beta^{1ij}} X\mathcal{F}(SCC(i,j,\psi,\varphi)) \wedge \mathcal{F}(\psi)$ $\wedge \neg\mathcal{F}(SCC(i,j,\psi,\varphi))$;
8. $\mathcal{F}(FuW(i,WCC(i,j,\psi,\varphi))) \stackrel{\mathcal{F}}{=} E_{\beta^{1ij}} X\mathcal{F}(WCC(i,j,\psi,\varphi)) \wedge \mathcal{F}(\varphi)$ $\wedge \neg\mathcal{F}(WCC(i,j,\psi,\varphi))$;
9. $\mathcal{F}(CaS(i,SCC(i,j,\psi,\varphi))) \stackrel{\mathcal{F}}{=} E_{\beta^{2ij}} X\mathcal{F}(SCC(i,j,\psi,\varphi)) \wedge \neg\mathcal{F}(\psi)$ $\wedge \neg\mathcal{F}(SCC(i,j,\psi,\varphi))$;
10. $\mathcal{F}(CaW(i,WCC(i,j,\psi,\varphi))) \stackrel{\mathcal{F}}{=} E_{\beta^{2ij}} X\mathcal{F}(WCC(i,j,\psi,\varphi)) \wedge \neg\mathcal{F}(\varphi)$ $\wedge \neg\mathcal{F}(WCC(i,j,\psi,\varphi))$;
11. $\mathcal{F}(ReS(j,SCC(i,j,\psi,\varphi))) \stackrel{\mathcal{F}}{=} E_{\beta^{3ji}} X\mathcal{F}(SCC(i,j,\psi,\varphi)) \wedge \neg\mathcal{F}(\psi)$ $\wedge \neg\mathcal{F}(SCC(i,j,\psi,\varphi))$;
12. $\mathcal{F}(ReW(j,WCC(i,j,\psi,\varphi))) \stackrel{\mathcal{F}}{=} E_{\beta^{3ji}} X\mathcal{F}(WCC(i,j,\psi,\varphi)) \wedge \neg\mathcal{F}(\varphi)$ $\wedge \neg\mathcal{F}(WCC(i,j,\psi,\varphi))$;
13. $\mathcal{F}(DeS(i,l,SCC(i,j,\psi,\varphi))) \stackrel{\mathcal{F}}{=} E_{\beta^{4ij}} X\mathcal{F}(SCC(i,j,\psi,\varphi)) \wedge \neg\mathcal{F}(\psi)$ $\wedge \neg\mathcal{F}(SCC(i,j,\psi,\varphi)) \wedge \mathcal{F}(SCC(l,j,\psi,\varphi))$;
14. $\mathcal{F}(DeW(i,l,WCC(i,j,\psi,\varphi))) \stackrel{\mathcal{F}}{=} E_{\beta^{4ij}} X\mathcal{F}(WCC(i,j,\psi,\varphi)) \wedge \neg\mathcal{F}(\varphi)$ $\wedge \neg\mathcal{F}(WCC(i,j,\psi,\varphi)) \wedge \mathcal{F}(WCC(l,j,\psi,\varphi))$;
15. $\mathcal{F}(AsS(j,l,SCC(i,j,\psi,\varphi))) \stackrel{\mathcal{F}}{=} E_{\beta^{5ij}} X\mathcal{F}(SCC(i,j,\psi,\varphi)) \wedge \neg\mathcal{F}(\psi)$ $\wedge \neg\mathcal{F}(SCC(i,j,\psi,\varphi)) \wedge \mathcal{F}(SCC(i,l,\psi,\varphi))$;
16. $\mathcal{F}(AsW(j,l,WCC(i,j,\psi,\varphi))) \stackrel{\mathcal{F}}{=} E_{\beta^{5ij}} X\mathcal{F}(WCC(i,j,\psi,\varphi)) \wedge \neg\mathcal{F}(\varphi)$ $\wedge \neg\mathcal{F}(WCC(i,j,\psi,\varphi)) \wedge \mathcal{F}(WCC(i,l,\psi,\varphi))$.

5.3.4 Soundness and completeness

Theorem 5.3.1 (Soundness and completeness of $\mathcal{F}$)**.** *Let M and φ be respectively a CTL$^{cc,\alpha}$ model and formula and let $\mathcal{F}(M)$ and $\mathcal{F}(\varphi)$ be the corresponding model and formula in ARCTL. We have $M \models \varphi$ iff $\mathcal{F}(M) \models \mathcal{F}(\varphi)$.*

The proof of this theorem is straightforward using induction with respect to the structure of the formula φ. Theorem 5.3.1 in principle proves that every CTL$^{cc,\alpha}$ formula has a corresponding ARCTL formula so that the CTL$^{cc,\alpha}$ formula holds in M iff the corresponding ARCTL formula holds in the ARCTL model obtained from M. It is worth mentioning

that we are discussing here the transformation (i.e., the reduction) of the model checking problem, which doesn't entail the equivalence of CTL$^{cc,\alpha}$ and ARCTL. To clarify this point, it is not the case that any model of ARCTL can correspond to a model in CTL$^{cc,\alpha}$. For instance, assume a model of ARCTL is obtained from a model of CTL$^{cc,\alpha}$ using the transformation function $\mathcal{F}$. Now, if we add a transition labeled by α^{ij} in the transformed ARCTL model, then the added transition doesn't necessarily correspond to a social accessibility relation in the model of CTL$^{cc,\alpha}$. The reason is because the social accessibility relation needs to satisfy the four conditions stated in Definition 3.4.1, which are not necessarily satisfied for the transition labeled by α^{ij} in the model of ARCTL. Consequently, it is possible, for instance, to have the formula $E_{\alpha^{ij}}X\mathcal{F}(\psi) \wedge A_{\alpha^{ij}}X(\mathcal{F}(\psi) \rightarrow \mathcal{F}(\varphi))$ satisfied in the model of ARCTL without having the corresponding formula $SCC(i, j, \psi, \varphi)$ satisfied in the model of CTL$^{cc,\alpha}$. Moreover, although the problem of model checking CTL$^{cc,\alpha}$ is reduced to the problem of model checking ARCTL, ARCTL is not more expressive than CTL$^{cc,\alpha}$. In fact, ARCTL cannot be used to reason about commitments and associated actions since the model of this logic doesn't support the accessibility relations. In the model of CTL$^{cc,\alpha}$, these relations are part of the model and are automatically computed based on the shared and unshared variables.

5.3.5 Space complexity

Theorem 5.3.2 (Lower bound)**.** *Let $\sqsubseteq_{lsr}$ denote the linear-space reduction. The problem of model checking CTL can be reduced into the problem of model checking CTL$^{cc,\alpha}$ in a linear space, i.e., MC(CTL) $\sqsubseteq_{lsr}$ MC(CTL$^{cc,\alpha}$).*

Proof. CTL$^{cc,\alpha}$ subsumes CTL because CTL$^{cc,\alpha}$ integrates CTL modalities, commitment modality and commitment action modalities. Accordingly, CTL formulae are CTL$^{cc,\alpha}$ formulae as well and models of CTL are part of CTL$^{cc,\alpha}$ models. The transformation of the model and the formula could be readily computed by a deterministic Turing Machine (TM) in a logarithmic space w.r.t. the size n of the CTL input model (i.e., space $O(\log n)$), and linear space w.r.t. the size of the CTL formula. For the model, TM simply looks at the input and writes in its output tape, one-by-one, the states (including the initial ones), valuation function, and transitions. For the formula, simply the same input CTL formula is produced as output. □

Theorem 5.3.3 (Upper bound)**.** *Let $\sqsubseteq_{psr}$ denote the polynomial-space reduction. The problem of model checking CTL$^{cc,\alpha}$ can be reduced into the problem of model checking ARCTL*

in a polynomial space, i.e., MC(CTL$^{cc,\alpha}$) $\sqsubseteq_{psr}$ MC(ARCTL).

Proof. The transformation of the CTL$^{cc,\alpha}$ model and the CTL$^{cc,\alpha}$ formula into the ARCTL model and the ARCTL formula could be readily computed by a deterministic Turing Machine (TM) in space $O(\log n)$ where n is the size of the input CTL$^{cc,\alpha}$ model, and polynomial space w.r.t. the size of the CTL$^{cc,\alpha}$ formula. For the model, TM reads in the input tape a model of CTL$^{cc,\alpha}$ and produces in the output tape, one-by-one, the same states including the initial ones and the same state valuation function. Moreover, TM writes α^0 and γ^{ik} in the set of actions Act if there are transitions defined in T and labeled transitions defined by R_{c_i}. It reads the accessibility relation $\sim_{i\to j}$ in the input model one-by-one and for each one, it writes α^{ij} in Act. TM also writes the set of actions β^{kij} in Act for symmetric transitions. Then, for each element in Act, TM writes in the output tape V_{Act} one-by-one as defined above. Finally, TM looks for all transitions (s, s') and $(l_i(s), a_i, l_i(s'))$ in the input model and writes, one-by-one, the transitions (g, α^0, g') and (g, γ^{ik}, g'). It also writes, one-by-one, the transitions (g, α^{ij}, g') for each accessibility relation $s \sim_{i\to j} s'$ in the input model, and transitions (g', β^{kij}, g) for the defined transitions (g, γ^{ik}, g') and (g, α^{ij}, g') in the output model. All these writing operations are clearly logarithmic in space because this transformation is done on-the-fly, step-by-step. We showed above that any formula of CTL$^{cc,\alpha}$ is transformable into a formula of ARCTL. All these transformations are clearly polynomial space in the length of the input formula, so the theorem. □

Theorem 5.3.4. *The space complexity of the CTL$^{cc,\alpha}$ model checking for concurrent programs is PSPACE-complete with respect to the size of the components P_i and the length of the formula.*

Proof. We proved in [52] that model checking GCTL* is PSPACE-complete for concurrent programs and also showcased that GCTL* subsumes ARCTL. Then, the upper bound of model checking ARCTL is PSAPCE. On the other hand, ARCTL subsumes CTL. Since model checking CTL is PSPACE-complete for concurrent programs [62, 77], the lower bound of model checking ARCTL is PSAPCE as well. We can conclude that model checking ARCTL is PSPACE-complete for concurrent programs. We also proved that MC(CTL) $\sqsubseteq_{lsr}$ MC(CTL$^{cc,\alpha}$) $\sqsubseteq_{psr}$ MC(ARCTL) in Theorems 5.3.2 and 5.3.3. The theorem follows from the fact that model checking CTL and ARCTL are both PSPACE-complete for concurrent programs with respect to the size of the components P_i and the length of the formula being checked. □

Chapter 6

Application Domains

This chapter[1] covers three application domains to address Question 1.3.8 mentioned in Chapter 1:

- Business interaction protocols.
- Health care processes.
- Web service compositions.

6.1 Introduction

As shown in Chapter 1, there is a huge variety of application areas for which agents technology is suitable [21]. Wooldridge divided applications of agents into two groups: 1) distributed systems, where multi-agents become processing nodes in a distributed fashion; and 2) personal software assistants, where individual agents proactively assist users to perform some actions (Chapter 11 in [98]). Clearly, interactions among processing nodes and among individual agents and their users is a key component to achieve individual objectives. In this case, all interactions take the form of a goal-driven behavior. Therefore, high-level interaction technologies are strongly needed, especially in open environments. The agent communication community introduced the theory of social commitments, not only to model such interactions, but also to engineer the topology of MASs in a flexible way [82, 105].

In this chapter, we investigate the application domains of our approach introduced in Chapter 3. It consists of two main components: conditional commitment specification language called CTL$^{cc,\alpha}$ and symbolic model checking implemented in the MCMAS+ tool

[1] The results in this chapter are collected from our publications in [39, 43, 44].

and its input language ISPL+. These components elegantly give the proposed approach a rigorous formal basis. To apply our approach, we first introduce a methodology to model MASs from given specifications. The methodology specifically focuses on modeling business relationships instead of low-level details based on creating and manipulating social commitments. This is because such relationships capture the dependencies among agents, while internal parts of each agent model local states including commitment states, local actions including commitment actions and local protocol (policy). Our methodology is summarized in Table 10. The methodology takes as input the formal or informal specifica-

Table 10: The steps of the proposed modeling methodology

Description	**Input**	**Output**
Identifying sub-scenarios	Formal or informal specification of MAS	Sub-scenarios
Identifying roles	Sub-scenario	Roles and their names
Identifying business tasks	Sub-scenario	Tasks
Introducing a commitment for each sub-scenario	Weak (or strong) commitment, sub-scenario, roles, tasks	Business commitment model
Introducing a commitment action for each commitment	Weak (or strong) commitment, business meaning of exchanged message, roles	Business commitment action model
Generating formal model	Specification of MAS, all commitment models and all commitment actions models	The whole formal model

tion of MAS in the form of business scenario. It works as follows:

1. It extracts all business interactions (sub-scenarios) from the system specification.
2. It identifies the roles and their names from each sub-scenario.
3. It identifies the business tasks carried out by the roles to achieve their interactive goals.
4. It creates a business commitment model for each sub-scenario to achieve the identified tasks with respect to the semantics of conditional commitments.
5. It creates a business commitment action model for each identified commitment with respect to the semantics of commitment action. The introduced action must match the business meaning extracted from exchanged message among roles.
6. It generates the whole formal model by formally combining the business models of all introduced commitments and their actions according to the given specification of

MAS and our model such that: 1) all redundant states and atomic propositions are removed; and 2) all created commitments are resolved. From the flexibility of social commitment-based approach, one commitment can precede another one as long as the behavior is correct at the business interaction level, such as the commitment to deliver requested goods can be created before the commitment to pay and vice versa.

Second, we define in Table 11 a library of properties/requirements that are required to

Table 11: Library of proposed properties

Property name	**Specification**
Reachability	There is a valid computation sequence to reach a particular state of interest
Safety	Something bad never happens
Liveness	Something good will eventually happen
Response	An event will happen infinitely many times
Guarantee	An event will eventually happen, but doesn't promise repetitions

be proved or verified on MASs modeled by our methodology using the developed tool. These properties are introduced in the literature of reactive and concurrent systems [69] and some of these properties are used in the literature of agent communication [51]. Through the adopted application domains, we present two forms of these properties: *simple* and *parametric*. The parametric form respects two interleaved techniques between interacting agents. Such properties are formalized using the CTL$^{cc,\alpha}$ logic.

Our long-term goals are to: 1) demonstrate the expressive power of CTL$^{cc,\alpha}$ in expressing the above requirements; 2) test experimentally the computational performance and scalability of the implemented tool; 3) confirm experimentally the theoretical complexity results; and 4) compare experimentally the proposed approach with other approaches (if any). We selected three significant application domains and performed a set of experiments to achieve these goals. The testbed of these experiments is as follows:

1. Desktop: Intel(R) Pentium(R) 4CPU @ 3.00GHz with 8GB of DDR3 RAM and 64-bit Operating System (Windows 7).
2. Eclipse with plug-in MCMAS's graphical user interface, Java JDK 8 and Graphviz 2.8.
3. The implemented tool MCMAS+.
4. Flex 2.5.4, GNU bison 2.3, GNU g++ 4.0.1 and Cygwin for 64-bit versions of Windows to compile tool.

6.2 First domain: Business interaction protocols

To make a dialogue or conversation, we need a mechanism to regulate and coordinate a set of ACL messages exchanged among participating agents. The developers of FIPA-ACL proposed a set of FIPA-ACL protocols[2]. These protocols include the standard FIPA messages, designed mainly for certain purposes. *Request Interaction protocol*, *English Auction Interaction protocol* and *Contract Net protocol* are the famous examples of FIPA-ACL protocols. The English Auction Interaction protocol is designed to acquire a higher market value for auction goods. The social approaches criticized the FIPA-ACL protocols as they restrict the protocols' flexibility [28, 51] by adding constraints on the occurrence of exchanged messages. Therefore, they are too rigid to be used by autonomous agents. A formalism of commitment machines has been used to model and execute multi-agent interaction protocols in a flexible and declarative way [104]. The resulting protocols are termed commitment-based protocols. In this formalism, the protocol states are labeled with commitments and literals holding on these states. A direct mapping from protocol messages (actions) into commitment actions is carried out through the modeling process. The transitions between protocol states are then labeled with commitment actions. To execute these machines, modeled protocols are compiled into finite state machines [104] or generalized Büchi automata [80]. However, the commitment machine formalism is unsuitable as it waives the: 1) formal description of agents participating in protocols; and 2) formal semantics of social commitments and their actions. To address these limitations, we formalized commitment-based protocols using our logical model, which formally includes the description of agents and defined computationally grounded semantics for social commitments and their actions. Hereafter, we describe the business interaction protocol called NetBill protocol, widely used in the literature of social commitments.

6.2.1 The NetBill protocol: Specification and modeling

The NetBill protocol is an electronic commerce protocol designed to be used for the selling and delivery of low-priced information goods such as software programs and journal articles over the Internet [84]. The protocol regulates the interactions among two agents called the merchant Mer and the customer Cus to build a MAS. According to our methodology, the protocol can be divided into four sub-scenarios: request, delivery, payment, and

[2]See FIPA-ACL Interaction Protocols (2001,2002), `http://www.fipa.org/repository/ips.php3`

receipt. In the request scenario, Cus asks Mer for a price quote for certain good items and Mer presents the price quote as an offer. Cus can accept or reject the offer. It is clear that there are only two roles named Cus and Mer and three tasks named $request$, $present$ and $accept$. The commitments to achieve these tasks are introduced latter (see $C4, C5$ and $C6$). If Cus accepts the received offer, the delivery scenario starts. In this scenario, there is no new roles and only one task named $deliver$, which needs to introduce a strong commitment to achieve it. From the dependency relation, the Mer role will play the debtor agent and the Cus role will play the creditor agent. The business model of this commitment has two social states connected by a transition labeled with 'deliverGoods' message. As the business meaning of the 'deliverGoods' message means fulfilling the introduced commitment, the methodology introduces a fulfill action and its business model, which consists of three social states. According to the semantics of fulfillment action, the first state should be commitment state, the second state should be fulfilled state and the third state is needed to terminate the active commitment. In the payment scenario, there is no new roles and only one task named $payment$, which needs to introduce a strong commitment to achieve it such that the Cus role will play the debtor agent and the Mer role will play the creditor agent. The business model of this commitment has two social states connected by a transition labeled with 'sendEPO' message, where EPO is the abbreviation of the electronic payment order. This message means fulfilling the introduced commitment and then the methodology introduces a fulfill action and its business model. In the receipt scenario, there is no new roles and only one task named $receipt$, which needs to introduce a strong commitment to achieve it such that the Mer role will play the debtor agent and the Cus role will play the creditor agent. The business model of this commitment has two social states connected by a transition labeled with 'sendReceipt' message. This message means fulfilling the introduced commitment and then the methodology introduces a fulfill action and its business model.

According to the NetBill protocol specification introduced in [84], our methodology generates the whole formal model, which combines the business models of all introduced commitments and their actions using our model $M = (S, T, \{\sim_{i\to j} \mid (i,j) \in \mathcal{A}^2\}, \{R_{c_i} \mid i \in \mathcal{A}\}, I, \mathcal{V})$. This model is illustrated in Figure 16 wherein the set of roles that will be instantiated by agents participating in the protocol is $\mathcal{A} = \{Cus, Mer\}$ plus an environment agent e to model the protocol itself. In the figure, instead of using a social transition, which captures the allowed evolution of the protocol from a social state to another social state, we used the corresponding local transitions, which are labeled by agents and their local actions

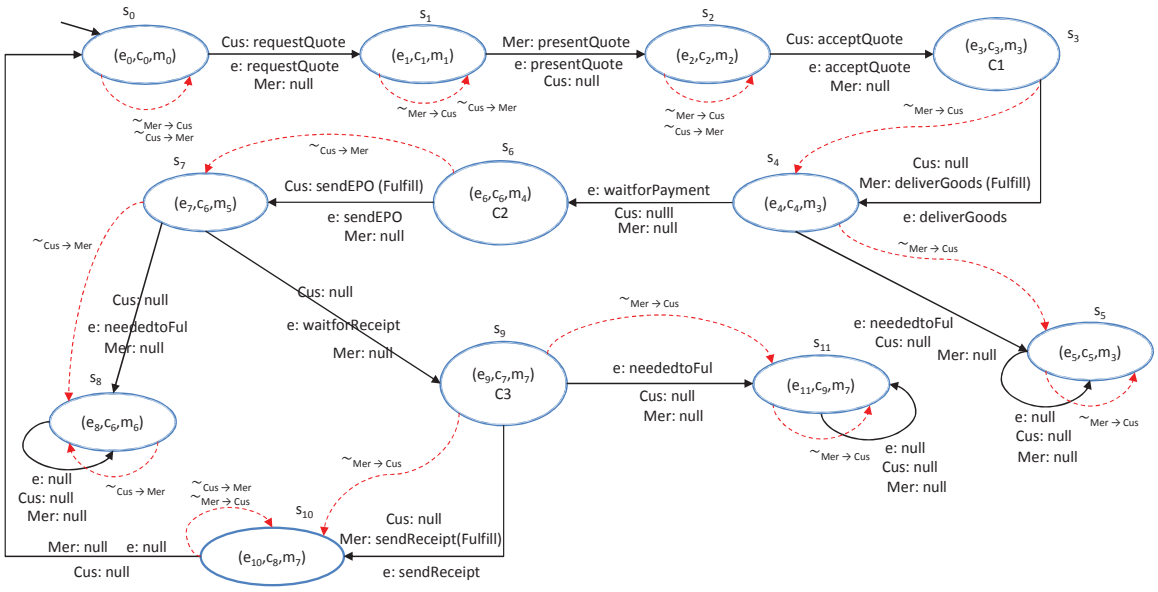

Figure 16: The graphical representation of our modeling of the NetBill protocol

performed during this social transition. For readability purpose, we used one transition instead of three local transitions. For example, the representative transition from s_0 to s_1 is labeled by: 1) Cus and its local action $requestQuote$; 2) Mer and its local action $null$; and 3) e and its local action $requestQuote$. Moreover, we added in parentheses the commitment action, which captures the meaning of the protocol message. The social states are numbered and identified with their local states of the three interacting agents. The dashed-red arrows refer to the social accessibility relations ($\sim_{Cus \to Mer}$ and $\sim_{Mer \to Cus}$) with respect to the active commitment. Some of states are labeled by the active commitments. Specifically, in the figure, there are three strong commitments constituting the protocol. They are defined as follows:

- $C1 = SCC(Mer, Cus, Quote(Accepted), AF\ Goods(Delivered))$
- $C2 = SCC(Cus, Mer, Goods(Accepted), AF\ Send(Payment))$
- $C3 = SCC(Mer, Cus, Payment(Received), AF\ Receipt(Delivered))$

The formula $C1$ expresses that the merchant strongly commits towards the customer to eventually deliver the requested goods when the customer accepts the received price quote. The formula $C2$ states that when the delivered goods are accepted by the customer, it will strongly commit to send the agreed payment to the merchant. The formula $C3$ means

that the merchant strongly commits towards the customer to eventually deliver the receipt solely when the payment is received. By modeling the protocol using strong commitments, the customer and merchant must respectively have the possibility of reaching states at which the propositions $Quote(Accepted)$, $Goods(Accepted)$, and $Payment(Received)$ are satisfied. The merchant can fulfill its commitments ($C1$ and $C3$) at states s_4 and s_{10} by respectively performing the actions 'deliverGoods' and 'sendReceipt', while the customer commitment ($C2$) can be fulfilled at state s_7 by performing the action 'sendEPO'. The interesting point here is that the life cycle of these strong commitments is implemented merely by considering fulfillment actions. This is semantically acceptable as strong commitments will be only active when their antecedents are true and, if so, the consequences must hold as well.

Our protocol modeling satisfies the regulative constraints introduced in [10, 9]. Indeed, the authors introduced the following three constraints: 1) a price quote must occur before the conditional commitment to pay ($C2$); 2) the conditional commitment to send a receipt ($C3$) must occur before the payment is done; and 3) if a receipt is issued, then a payment must have occurred before. On the other hand, the authors in [105, 97, 80] showed that the commitment-based approach flexibly enhances the NetBill protocol specification by ensuring that agents should be allowed to adjust their actions so as to handle exceptions that arise during interaction. Three of these exceptions—discussed in Example 3.2.1—would be expressed by the following strong commitments:

- $C4 = SCC(Mer, Cus, \top, Quote(Presented))$
- $C5 = SCC(Cus, Mer, \top, E(\neg Quote(Requested)\ U\ \neg Quote(Requested) \wedge Quote(Accepted)))$
- $C6 = SCC(Mer, Cus, \top, E(\neg Quote(Accepted)\ U\ \neg Quote(Accepted) \wedge Goods(Delivered)))$

Here, $C4$ means that the merchant proactively presents a quote even without being requested by the customer (mimicking the idea of advertising a price); $C5$ means that there is a possibility for the customer to accept the price quote without requesting it (as in cases of trust); and $C6$ states that the merchant strongly commits to deliver the goods without asking the customer for accepting the price (as in a trial offer). By considering each exception as a different instance of the protocol, then we have four protocol instances including the instance of the protocol itself. Other exceptions are: 1) the customer can reject the presented price quote and the protocol passes through the initial state; 2) the customer can

send the payment before initially receiving its requested goods; 3) the customer can pay by the credit instead of sending the electronic payment order; 4) the agents can detach conditional commitments into unconditional commitments; and so forth. In total, we have 15 instances of the protocol in which each customer is working with a merchant within a protocol instance. The key motivations of considering different instances are that: 1) potential customers and merchants will behave and execute the protocol in different ways; 2) several types of business scenarios might coexist; and 3) the scalability of the modeled system can be defined in an attractive way. Having different combinations allow us to achieve different levels of scalability that makes the problem complex enough to observe significant results and to compare related proposals.

6.2.2 Protocol properties

Various protocol properties are expressed in the CTL$^{cc,\alpha}$ logic to verify the NetBill protocol specification.

1. Reachability property. The following lists the formulae that can be exploited to check the reachable states in the NetBill protocol:
   - $\varphi_1 = E\big(\neg Quote(Received)\ U\ (Quote(Received) \wedge SCC(Mer, Cus, Quote(Accepted), AF\ Goods(Delivered)))\big)$
   - $\varphi_2 = E\big(\neg Goods(Delivered)\ U\ (Goods(Delivered) \wedge SCC(Cus, Mer, Goods(Accepted), AF\ Send(Payment)))\big)$
   - $\varphi_3 = E(\neg Send(Payment)\ U\ (Send(Payment) \wedge Receipt(Delivered)))$
   - $\varphi_4 = EF\ \big(FuS(Mer, SCC(Mer, Cus, \top, Quote(Presented)))\big)$
   - $\varphi_5 = EF\ \big(FuS(Cus, SCC(Cus, Mer, \top, E(\neg Quote(Requested)\ U\ \neg Quote(Requested) \wedge Quote(Accepted)))))\big)$
   - $\varphi_6 = EF\ \big(FuS(Mer, SCC(Mer, Cus, \top, E(\neg Quote(Accepted)\ U\ \neg Quote(Accepted) \wedge Goods(Delivered)))))\big)$

   For example, the formula φ_1 means that there exists a path where the merchant will not strongly commit to deliver the requested goods to the customer until the customer received and accepted its price quote.

2. Safety property. For example, a bad situation is that the customer sends the payment for the requested goods, but the merchant never strongly commits to deliver the receipt:

   $\varphi_7 = AG\ \neg\big(Send(Payment) \wedge \neg SCC(Mer, Cus, Payment(Recieved), AF\ Receipt(Delivered))\big)$

3. Liveness property. For example, along all paths globally if the merchant delivers the requested goods, then there is a path in its future the customer will strongly commit to send the payment when it accepts the delivered goods:

$\varphi_8 = AG\big(Goods(Delivered) \rightarrow EF(SCC(Cus, Mer, Goods(Accepted), AF\ Send(Payment))))$

6.2.3 Experimental results of the NetBill protocol

To encode the NetBill protocol described in Section 6.2.1 and formalized using our temporal model, we used the automated ISPL+ model generated by MCMAS+. In particular, we exploited the following sections `Agent Environment ... end Agent`, `Agent Cus ... end Agent`, and `Agent Mer ... end Agent` to encode the protocol specification itself, and the roles played by the customer and merchant agents, respectively. The interesting point in the agent section is the set of shared variables, which gives the agent the possibility to directly communicate with other agents. For example, in our ISPL+ encoding of the customer and merchant agents, we define the set $x\_1 = \{x_0, x_1\}$ of shared variables to establish a communication channel between them. The value of $x\_1 = x_0$ for the customer at the local state $c3$ is changed into $x\_1 = x_1$ at the local state $c4$ to make the social state $s_4 = (e4, c4, m3)$ accessible from the social state $s_3 = (e3, c3, m3)$. According to the first condition in Definition 3.4.1, the local state $m3$ of the merchant doesn't change as it is the debtor agent. The set of atomic propositions is added into the `Evaluation ... end Evaluation` section, while the set of initial states is added into the `InitStates ... end InitStates` section. Furthermore, the defined properties in Section 6.2.2 are encoded using ISPL+ and inserted into the `Formulae ... end Formulae` section at the end of the model. Having encoded the protocol, we proceeded to verify its correctness against the defined formulae using the developed MCMAS+ tool. The customer and merchant agents are specified using ISPL+ as follows:

```
Agent Cus
  Vars:
    x_1: {x0, x1};//The set of shared variables
    z_1: {z0, z1};//The set of unshared variables
    c_1: {c0,c1,c2,c3,c4,c5,c6,...};
  end Vars
  Actions = {c_requestQuote,c_acceptQuote,...,c_null};
  Protocol:
    c_1=c0: {c_requestQuote};
    c_1=c2: {c_acceptQuote};
    ...;
    Other: {c_null};
  end Protocol
```

```
  Evolution:
    c_1=c1 if c_1=c0 and Action=c_requestQuote and
    Env.Action=e_requestQuote;
    c_1=c2 if c_1=c1 and Mer.Action=m_presentQuote
    and Env.Action=e_presentQuote;
    c_1=c3 and x_1=x0 if c_1=c2 and Action=c_acceptQuote and
    Env.Action=e_acceptQuote;
    c_1=c4 and x_1=x1 and z_1=z0 if c_1=c3 and Mer.Action=Fulfill_Mer
    and Env.Action=e_deliverGoods;
    ...;
    c_1=c9 and x_1=x1 and z_1=z1 if c_1=c8 and Env.Action=
    e_neededtoFulCom2;
    c_1=c0 and x_1=x0 and z_1=z0 if c_1=c8 and Env.Action=e_null;
  end Evolution
end Agent
Agent Mer
  Vars:
    x_1: {x0,x1};//The set of shared variables
    y_1: {y0,y1};//The set of unshared variables
    m_1: {m0,m1,m2,m3,m4,m5,...};
  end Vars
  Actions = {m_presentQuote,Fulfill_Mer,m_sendReceipt,...,m_null};
  Protocol:
    m_1=m1: {m_presentQuote};
    m_1=m3: {Fufill_Mer};
    ...;
    Other: {m_null};
  end Protocol
  Evolution:
    m_1=m1 if m_1=m0 and Cus.Action=c_requestQuote and Env.
    Action=e_requestQuote;
    m_1=m2 if m_1=m1 and Action=m_presentQuote and
    Env.Action=e_presentQuote;
    m_1=m3 and x_1=x1 and y_1=y0 if m_1=m2
    and Cus.Action=c_acceptQuote and
    Env.Action=e_acceptQuote;
    ...;
    m_1=m7 and x_1=x1 and y_1=y1 if m_1=m5 and Env.Action=
    e_waitforReceipt;
    m_1=m0 and x_1=x1 and y_1=y0 if m_1=m7 and Env.Action=e_null;
  end Evolution
```

```
end Agent
```

We performed a set of experiments and from Experiment 2 we rewrite the defined formulae in a parametric form. For example, in Experiment 5 the formula φ_1 is redefined as follows:

$$\varphi_1' = \bigwedge_{i=1}^{10} E\big(\neg Quote(Received)_i \; U \; (Quote(Received)_i \; \wedge SCC(Mer_i, Cus_i, \\ Quote(Accepted)_i, AF \; Goods(Delivered)_i))\big)$$

We found that our 8 tested formulae (reachability, liveness, and safety) are satisfied in all these experiments. With regard to the Propositions 3.5.1 and 3.5.2, the corresponding weak commitments and their fulfillments of the strong ones are satisfied as well. Therefore, we do not need to test weak formulae.

6.2.4 Analyzing and evaluating results

To check the computational performance and scalability of the developed tool, we reported 6 experiments in Table 12 where the number of agents (#Agents), the number of reachable states (#States), the execution time (Time(Sec)) in seconds, and the memory in use (Memory(MB)) in megabytes are given. Our system of agents ranges from 6 agents (2 customers, 2 merchants, and 2 protocol instances modeled as 2 environment agents) to 45 agents (15 customers, 15 merchants, and 15 environment agents). From Table 12, the number of reachable states reflects the state space increases exponentially when the number of agents increases as we expected. However, the memory usage increases merely polynomially, which confirms the theoretical result (i.e., the PSPACE-completeness). With regard to the execution time, the increase is not exponential, but faster than the polynomial. Although the verification process takes more time, it is acceptable for detecting design errors for at least two crucial reasons. The former reason is about reducing the cost and time

Table 12: Verification results of the NetBill protocol

#Agents	#States	Time(Sec)	Memory(MB)
6	144	0.041	9
12	20736	0.365	13
18	2.98598e+06	1.875	23
24	4.29982e+08	6.892	46
30	6.19174e+10	19.258	66
...	...	...	...
45	1.5407e+16	46807.400	3181

requirements needed for both implementation and maintenance phases. The latter reason is in avoiding catastrophic situations in terms of both human lives loss and economic damages (think of failing safety property in control systems of nuclear power plants).

We continue evaluating the effectiveness of the developed tool with respect to complex models, which enable each agent to interact with all other agents in the system in a fully interleaved way. To implement this kind of the interaction technique, we used our encoding of the protocol model introduced above and then defined two parameterized tested formulae in which every customer interacts with every merchant, and vice versa.

$$\psi_1 = \bigwedge_{i=1}^{m} EF\big(\bigwedge_{j=1}^{n} SCC(Mer_i, Cus_j, Quote(Accepted), AF\ Goods(Delivered))\big)$$

$$\psi_2 = \bigwedge_{i=1}^{m} EF\big(\bigwedge_{j=1}^{n} FuS(Mer_i, SCC(Mer_i, Cus_j, Quote(Accepted), AF\ Goods(Delivered)))\big)$$

Here, m and n refer to the number of merchants and customers, respectively. The formula ψ_1 says that there is a path in its future each merchant from m identified merchants can strongly commit to n customers to eventually deliver the requested goods when each customer from these identified customers accepts the received price quote. The formula ψ_2 is the fulfillment of this commitment. As a benchmark, we redefined ψ_1 and ψ_2 in the previous parametric form and called them ψ_3 and ψ_4 in which each customer is paired with a merchant and all these pairs evolve in a parallel way. We call this kind of interaction a "pure interleaved interaction".

$$\psi_3 = \bigwedge_{i=1}^{m} EF\ SCC(Mer_i, Cus_i, Quote(Accepted), AF\ Goods(Delivered))$$

$$\psi_4 = \bigwedge_{i=1}^{m} EF\ FuS(Mer_i, SCC(Mer_i, Cus_i, Quote(Accepted), AF\ Goods(Delivered)))$$

Table 13 reports the verification results of four experiments. We found that all tested formulae are satisfied. Since we utilized our previous encoding, the number of reachable states is similar to the corresponding one in Table 12. However, since here we use different testing formulae, the memory usage results differ from the corresponding ones in Table 12. From Table 13, the memory usage in the fully interleaved interactions, as we expected, is higher than the corresponding ones in the pure interleaved interactions, because there are more interactions among agents. This enforces that the MCMAS+ tool keep more states

Table 13: Verification results of the NetBill protocol against ψ_1, ψ_2, ψ_3, and ψ_4

		Pure Interleaved Interactions against ψ_3 and ψ_4	**Fully Interleaved Interactions** against ψ_1 and ψ_2
#Agents	**#States**	**Memory(MB)**	**Memory(MB)**
6	144	7	7
12	20736	9	11
18	2.98598e+06	21	25
24	4.29982e+08	71	92

until the established interactions will terminate. Figure 17 illustrates the relationship between the number of agents and memory usage in fully and pure interleaved interactions. We employed the standard polynomial trend to calculate the polynomial function and then

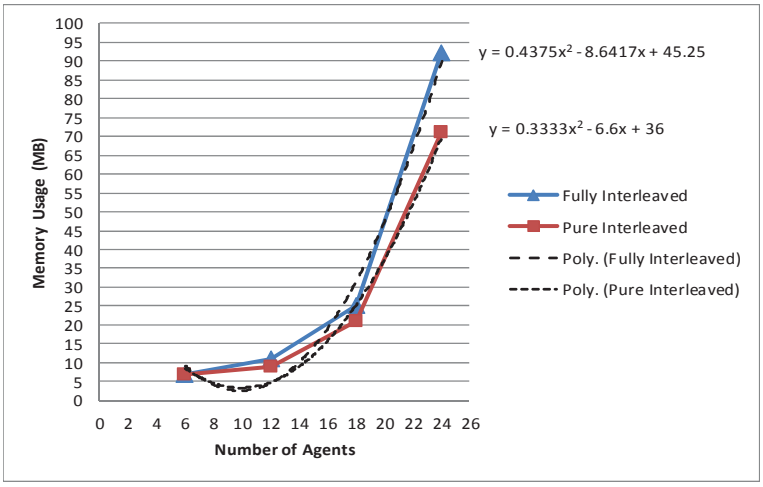

Figure 17: The relationship between the number of agents and memory usage

drew the polynomial curve that best fits our results based on the number of known X and Y values. We found the variation that deviates from the standard curve and our curve is very small at the first two experiments and zero in the resting again confirm the theoretical results (i.e., the PSPACE-completeness). The calculated polynomial functions can be used to estimate, for instance, the future value of the memory usage, given the number of agents or to in-fill results (either the memory usage or the number of agents) where gaps might be present in the obtained results. To this end, such results certify the effectiveness of the developed algorithm.

6.2.5 Comparing results

In order to compare the proposed technique with existing techniques, Table 14 reports the time and memory results obtained in the verification of the NetBill protocol respectively using direct and indirect techniques presented in Chapter 1. From the table, there is more

Table 14: Comparison between current direct and indirect verification techniques

Direct Technique			**Indirect Techniques**		
El Menshawy et al. [53]			El Menshawy et al. [52]		
#Agents	**Time(Sec)**	**Memory(MB)**	**#Agents**	**Time(Sec)**	**Memory(MB)**
2	< 0.01	8.600	2	0.020	4.241
4	< 0.01	8.971	3	0.184	5.507
6	< 0.01	9.958	4	2.736	12.957
8	< 0.01	12.056	5	63.687	15.432
10	1.00	16.856	6	630.914	83.839
12	2.00	36.134	El Menshawy et al. [54]		
14	8.00	45.592	2	0.190	
16	29.00	56.280	3	0.746	
18	426.00	94.360	4	6.635	
20	1128.00	153.008	5	32.860	
			6	286.932	
			7	1684.409	
			8	5123.356	
			Bentahar et al. [16]		
			2	0.5	
			El Menshawy et al. [49]		
			4	2	

overhead in terms of the time [52, 54] and memory [52] in the indirect techniques compared to the direct technique [53] (see, for instance, when the number of agents is 6 agents), due to the transformation process of the model and formulae before the actual verification process is undertaken. Furthermore, the NetBill protocol was model checked against some properties expressed in an extension of CTL* using indirect techniques introduced by [16] and [49], but they solely considered 2, and 4 agents, respectively. Figure 18 shows the relationship between the number of agents and execution time using indirect [52, 54] and direct [53] techniques as well as the current technique. For the purpose of simplicity, in Figure 18, we considered merely 30 agents in the current technique without loss of generality as the maximum number of agents in other techniques is 20 agents. From Figure

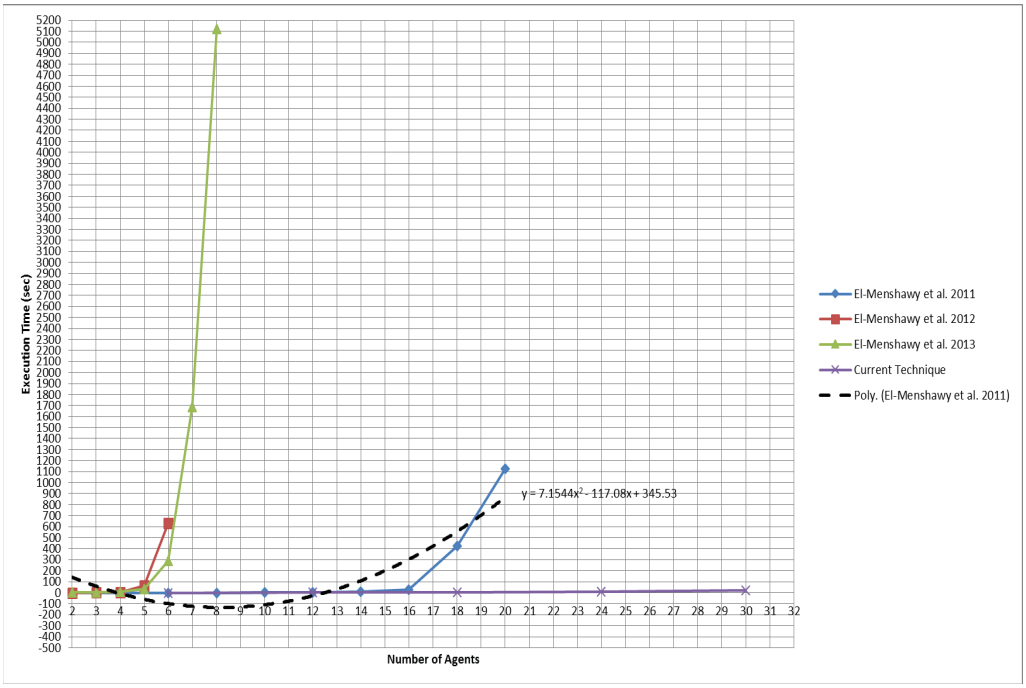

Figure 18: The relationship between the number of agents and execution times

18, we can readily observe that the current technique takes less time to complete the verification process than other techniques. Furthermore, the calculated polynomial function ($Y = 7.1544X^2 - 117.08X + 345.53$) that best fits the execution time series of the direct technique introduced in [53] is utilized to compare the effectiveness of this technique with the proposed direct technique. The idea is to compute the execution times as predication values (i.e., they are not experimentally computed in [53]) so as to compare with the corresponding experimentally computed ones in the proposed technique, given the number of agents. For example, $Y = 1656.54sec$ and $Y = 2581.03sec$ when $X = 24$ and $X = 30$ agents, respectively. It is obvious that these values are far from the corresponding ones reported in Table 12. Furthermore, the memory consumed in our technique (cf. Table 12) is less than the corresponding one consumed in other techniques (cf. Table 14). We conclude that the computational performance and scalability of the developed algorithm implemented in our tool is better than the one developed in [53] in terms of the execution time, memory usage, and number of agents (45 agents).

6.3 Second domain: Health care processes

Health care is the process of improving health by different methods, such as treatment, diagnosis, and prevention of diseases. Health care systems consist of multiple individuals and organizations introduced to achieve the health needs. Such systems are naturally characterized by distributed decisions and management of patient care, which needs complex interactions among various participants, such as physicians, medical specialists, hospitals, laboratories, insurance providers and patients. The paradigm of MASs was introduced into the health care domains to easy the management of taking required decisions and actions and to insure the existence of communication and coordination among participants [22]. Also, MASs support the heterogeneity and autonomy of the participants. In the literature, there are several medical applications that are implemented using The MASs paradigm (see, for example, [19, 60]). Recently, the authors in [91] introduced a business methodology to model health care processes using creation and manipulation of social commitments. They empirically showed that this methodology is superior to a traditional approach based on the Health Level Seven Messaging Standard[3] in "flexibility and objective quality", which is significantly validated by student's t-test. This flexibility aspect results from using a social commitment-based approach. The authors also claimed that an existing work that uses the reference information model (RIM) and unified medical language system (UMLS) concentrates only on operational models that particularly tend to hide business relationships. Hiding business relationships makes understanding why participants exchange particular messages and why these messages are ordered in a particular way difficult.

In this application, we are not only modeling health care processes using conditional commitments, but also using conditional commitment actions. A computationally grounded semantics opens the way to automatically verify the correctness of the modeled process. This verification process is urgently needed, since incorrect modelings can conduct to inefficiencies and to critical medical errors basically impacting patient safety.

6.3.1 The breast cancer diagnosis and treatment process: Specification and modeling

We claim that our approach can efficiently advance the state of the art, which advocates the MASs paradigm to model the health care process. To support our claim, we considered the

[3] http://www.iso.org/iso/catalogue_detail.htm?csnumber=44428

health care process of breast cancer diagnosis and treatment (BCDT), introduced in the Assistant Secretary for Planning and Evaluation (ASPE) project, U.S. Department of Health and Human Services[4]. Specifically, the health care process of BCDT begins when a patient visits "a primary care physician", who notices a suspicious mass in the patient's breast. The physician then sends the patient to a radiologist to do a mammography (a diagnostic procedure to detect breast tumors by the use of X-rays). When the radiologist observes suspicious calcifications, he will send a report to the physician recommending a biopsy (also called an examination of tissues). In this condition, the physician instantly requests a radiologist to carry out a biopsy. The radiologist first collects a tissue specimen from the patient, varying from image-guided needle biopsies to large surgical specimens, and second sends the tissue specimen to a pathologist along with pertinent clinical information. The assigned pathologist analyzes the tissue specimen by direct macroscopic and microscopic examination of tissues, and carries out additional studies (e.g., molecular pathology) to render a definitive diagnosis of cancer. The pathology and radiologist often need a conference to accommodate their findings (results) and then generate a consensus report. The physician finally reviews the complete report with the patient to assign a treatment plan. On the other hand, the pathologist usually forwards his report to a registrar that should insert the patients name into a cancer registry.

By applying our methodology, we divided the BCDT process into 7 sub-scenarios: examination request, mammography refer, biopsy recommendation, collection and analysis tissue, report, treatment plan and registration. These sub-scenarios are played by five roles named: patient ($Patie$), physician ($Physi$), radiologist ($Radio$), pathologist ($Patho$), and registrar ($Regis$). We also identified a set of 12 tasks and introduced the appropriate conditional commitments based on identified dependency relations to achieve 6 consequence tasks when 6 antecedent tasks are met. The names of these tasks will be introduced in Section 6.3.2. Then, we generated the business models of the introduced commitments. Finally, we introduced the fulfillment actions capturing the business meanings of exchanged messages and generated their business models. Similar to the previous case study, we used the aforementioned process specification to generate the whole formal model, which combines all conditional commitments and their fulfillments models using our model $M = (S, T, \{\sim_{i\to j} \mid (i,\, j) \in \mathcal{A}^2\}, \{R_{c_i} |\, i \in \mathcal{A}\}, I, \mathcal{V})$. Thus, the set of roles that will be instantiated by agents participating in the process is $\mathcal{A} = \{Patie,$

[4]Available at `http://aspe.hhs.gov/sp/reports/2010/PathRad/index.shtml`.

$Physi, Radio, Patho, Regis\}$ plus an environment agent e to model the BCDT process itself.

6.3.2 Process properties

The following CTL$^{cc,\alpha}$ formulae express the discussed requirements in the BCDT process, where $\xi \in \{S, W\}$:

- $\varphi_1 = AG(Examination(Requested) \rightarrow EF\ \xi CC(Physi, Patie, Mass(Detected), AF\ Mammography(Refered)))$
- $\varphi_2 = EF\ Fu\xi(Physi, \xi CC(Physi, Patie, Mass(Detected), AF\ Mammography(Refered)))$
- $\varphi_3 = EF\ \xi CC(Radio, Physi, Calcifiction(Detected), AF\ Biopsy(Recommended))$
- $\varphi_4 = EF\ Fu\xi(Radio, \xi CC(Radio, Physi, Calcifiction(Detected), AF\ Biopsy(Recommended)))$
- $\varphi_5 = EF\ \xi CC(Radio, Physi, Biopsy(Requested), AF\ Tissue(Collected))$
- $\varphi_6 = EF\ Fu\xi(Radio, \xi CC(Radio, Physi, Biopsy(Requested), AF\ Tissue(Collected)))$
- $\varphi_7 = EF\ \xi CC(Patho, Radio, Tissue(Received), AF\ Tissue(Analyzed))$
- $\varphi_8 = EF\ Fu\xi(Patho, \xi CC(Patho, Radio, Tissue(Received), AF\ Tissue(Analyzed)))$
- $\varphi_9 = AG(Results(Accommodated) \rightarrow EF\ \xi CC(Physi, Patie, ConsensusReport(Received), AF\ TreatmentPlan(Argeed)))$
- $\varphi_{10} = EF\ Fu\xi(Physi, \xi CC(Physi, Patie, ConsensusReport(Received), AF\ TreatmentPlan(Argeed)))$
- $\varphi_{11} = EF\ \xi CC(Regis, Patho, Report(Received), AF\ PatientName(Added))$
- $\varphi_{12} = EF\ Fu\xi(Regis, \xi CC(Regis, Patho, Report(Received), AF\ PatientName(Added)))$

It is obvious that such formulae are of sort reachability and liveness properties. For example, φ_1 states that whenever the patient requests an examination by the physician, then there exists a possibility for the latter to strongly (or weakly) commit to eventually refer the patient to a radiologist for a mammography once he detects a suspicious mass, and φ_2 expresses that there exists a path in its future the commitment in φ_1 will be fulfilled. These formulae are indeed exploited to check the correctness of the process model.

6.3.3 Experimental results of the breast cancer diagnosis and treatment process

As we did in the previous case study, we used our automated ISPL+ model generated by MCMAS+ to encode the process model and the defined formulae in Section 6.3.2 wherein each agent in the model is characterized by a set of local states, a set of shared variables, a set of local actions, a local protocol, etc. For example, the physician agent is specified using ISPL+ as follows:

```
Agent Physi
Vars:
ph:{ph0,ph1,ph2,ph3,ph4,...};
a:{a0,a2,a3};//to establish communication channel with patient
c:{c0,c1,c2,c3};//to establish communication channel with radiologist
d:{d0,d1,d2};//the set of unshared variable
end Vars
Actions = {physi_doExamination,Fulfill_Physi,physi_requestBiopsy,
...,physi_null};
Protocol:
ph=ph1:{physi_doExamination};
ph=ph2:{Fulfill_Physi};
...;
Other:{physi_null};
end Protocol
Evolution:
ph=ph1 if ph=ph0 and Patia.Action=patia_requestforExamen
and Env.Action=e_requestforExamination;
ph=ph2 and a=a2 if ph=ph1 and Action=physi_doExamination
and Env.Action=e_requestforExamen;
ph=ph3 and c=c0 and d=d1 if ph=ph2 and
Action=Fulfill_Physi and Patia.Action=patia_presentReferent
and Env.Action=e_presentReferent;
...
ph=ph9 and a=a3 if ph=ph7 and Patho.Action=path_sendReporttoPhysi
and Env.Action=e_sendReporttoPhysi;
end Evolution
end Agent
```

Table 15 reports the verification results of the process model against the above formulae within 10 experiments.

6.3.4 Analyzing and evaluating results

We found that all tested formulae are satisfied. In a matter of fact, from Experiment 2 we: 1) rewrite the defined formulae in a parametric form as we did in the previous case study using the pure interleaved technique; and 2) add one patient and an instance of the BCDT process; so in the last experiment, we have 10 patients. From the table, the state

Table 15: Verification results of the health care process of BCDT

#Agents	#States	Time(Sec)	Memory(MB)
6	21	0.277	6.80
8	119	0.165	7.80
10	711	0.487	9.80
12	4259	0.609	9.20
14	25431	0.724	10.90
16	151499	1.206	11.20
18	901551	2.084	18.10
20	5.36394e+06	2.099	17.00
22	3.19241e+07	2.446	19.10
24	1.90116e+08	3.46	19.09

space increases exponentially when the number of agents increases. On the other side, the increment in the memory usage is not linear but fairly close to polynomial, which confirms

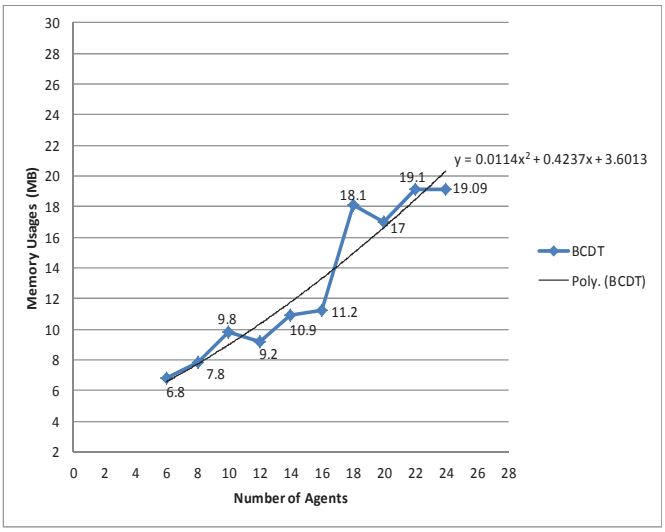

Figure 19: The relationship between the number of agents and memory usage

our theoretical results. For example, the memory usage Y using the calculated polynomial

function $Y = 0.0114X^2 + 0.4237X + 3.6013$ is $18.4\ MB$ when the number of agents is 22 (i.e., $X = 22$), while the experimental memory-usage is $19.1\ MB$ (cf. Table 15). The standard polynomial trend along with the calculated polynomial function are depicted in Figure 19.

6.4 Third domain: Web service compositions

6.4.1 Introduction and challenging issues

Web service composition is the capability to aggregate multiple services into a composite service to actually promote a certain functionality that would not have seemingly been possible by a single service (e.g., Expedia aggregates hotel reservation, car rental, and airline booking services). Interaction is the key foundation of composite services and service-oriented architecture (SOA), which provides a conceptual model for understanding and implementing web services and relationships among the components of this model. This is because valuable composite services will emerge from the interaction of more specialized services, and the SOA architecture is based on the interaction among three different components (service provider, service requester and service registry) to publish, invoke and register services. The importance of the interaction is made clear through practical and industrial notions of modeling composite services [13]: *orchestration* and *choreography*. Orchestration describes how multiple services can interact with each other from the perspective of a single participant using invocation-based approaches. This single participant acts as an orchestrator, coordinating the invoked services and linking the results computed by them. It also uses traditional programming constructs (such as loops, conditional branches, forks, and joins) to handle data received/sent from/to partner web services. Furthermore, orchestration refers to automated execution of a workflow in the sense that it produces executable business processes that could interact with internal and external services. Since orchestration doesn't describe a coordinated set of interactions between partners, but rather the execution of a specific business process, it only provides a local view. This local view of orchestration is not sufficient to specify interaction protocols; instead a global view of the interaction among participating web services has been acknowledged [12, 27]. This level of specification is captured by the notion of choreography, which particularly imposes a legal sequence of messages exchanged among participants. The business process execution language for web services (WS-BPEL) [1] is an orchestration language. Examples of choreography languages include web service choreography interface (WSCI) [5]

and web services choreography description language (WS-CDL) [13]. Differently from orchestration, the focus of choreography is not on generating executable business processes, but rather on specifying the public contracts that supply the necessary rules of business engagements for making all the interacting participants correctly cooperate and collaborate. Figure 20 graphically shows the relationship between orchestration and choreography (adapted from [74]).

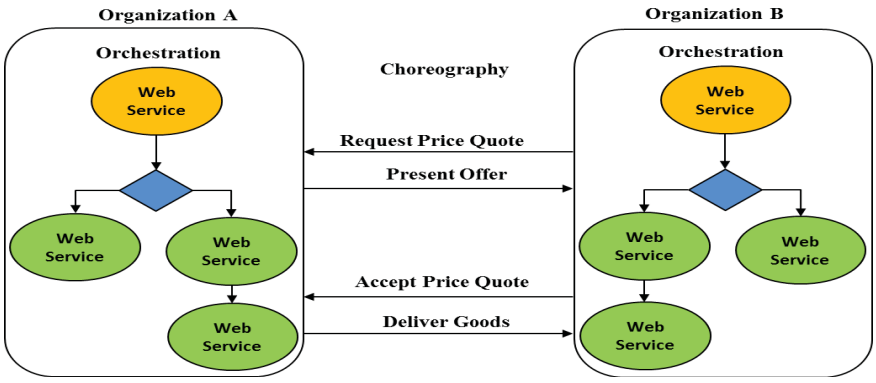

Figure 20: Orchestration vs choreography in modeling composite ervices

Although the technology for developing basic services has attained a certain level of maturity, there remain open challenges when it comes to engineering composite services. Those challenges concern: 1) the engagement in modeling complex business interactions that go beyond simple sequences of ordered messages from high-level abstractions; 2) the specification of interaction protocols that characterize the global behaviors of web services composition and regulate the interactions among them in a declarative manner; and 3) the automatic detection of undesirable service behaviors, given interaction protocols (i.e., how can we be certain the systems of composite services will be safe and reliable?).

6.4.2 The proposed approach

Our aim is to propose an approach to address the aforementioned challenges in an integrated framework. The proposed approach specifically acquires the experience of the research community that adopts: 1) the paradigm of MASs; and 2) the symbolic model checking techniques. By traversing across these domains, we aim to capture the benefits of their cross-fertilization, which in principle demonstrate concerning convergence points.

Roughly speaking, MAS strongly provides a mainstream framework for modeling and

reasoning about services and their compositions [65]. In fact, the relationship between agents and services is defined by the W3C consortium [20] as follows: "a web service is an abstract notion that must be implemented by a concrete agent. The agent is the concrete piece of software that sends and receives messages". We capture this relationship by calling agent-based web service. Furthermore, MAS theories are appropriate in the consideration of complicated web service interactions and directly support a rich, flexible, and expressive range of specifications. Concretely speaking, we utilize multi-agent social commitments and a special type of multi-agent interaction protocols, namely commitment-based protocols, which are now widely recognized as a powerful representation and regulation for the interactions among autonomous and heterogeneous agents [9, 15, 54, 79].

A choreography supported by WS-CDL (web services choreography description language) [13] often specifies and constructs systems that could closely agree with the manner of constructing MASs. However, it is not suitable for an agent communication theory [27]. This is because there is no a clear manner to define the meanings of exchanged messages, which are precisely needed to enable an agent to carry out high-level reasoning about protocol actions, and to interoperate effectively with other agents. The interoperability readily means all exchanged messages among agents are understood by each other and their interaction is deadlock-free. Also, WS-CDL adds certain actions into a choreography to guide a participating service, for example, to decide what it ought to do when receiving a message [27], but these actions are invariably private.

While commitment-based protocols can play the same role of choreographes, they define the meanings of the interactions in terms of social commitments and commitment actions. Thus, the engineering of composed MASs means rigorously and declaratively specifying the interactions among multi-agent based web services by the way of commitment-based protocols. According to a declarative specification, commitment-based protocols are flexible, making them suitable for accommodating dynamic changes or exceptions [27, 105], and can be directly verified [54, 94] without requiring the transformation into other formalisms (e.g., Petri nets).

Our approach supports a dynamic service composition using commitment actions—especially assignment and delegation actions—regulated by the same protocol and/or different protocols that can be automatically verified against certain composition specifications using our symbolic model checker. Such specifications are expressed in CTL$^{cc,\alpha}$. Based on these reasons, we associate our models to commitment-based protocols to address the limitations of choreographies. Symbolic model checking [32] is one of the most

promising techniques, which help designers of multi-agent based web services detect and then eliminate or repair undesirable service behaviors to guarantee an ideal outcome of the composite services [65]. On the other hand, orchestrations can be seen as a special kind of our improved choreographies in which all interactions are made between the orchestrating service and one of the partners and no interaction is considered among partners.

6.4.3 Formal specification language of composing contract protocols

Business interactions are typically defined by contracts that describe the roles and responsibilities of their parties along with a list of failure conditions and associated penalties. Most industrial use cases involving contracts are currently analyzed by considering the contract compliant scenarios, described by transition systems [65] without investigating the correctness of contracts themselves among multi-agents based web services that are often ambiguous and error prone [38]. Following Desai et al. [38], we have two types of contracts: simple and complex. A simple contract can be viewed as a social commitment among its parties, while a collection of simple contracts (or social commitments) forms complex contract. Thus, we understand the interactions via exchanging messages that would occur within the execution of a simple business contract in terms of how they affect the states of commitment. Differently from modeling commitments as fluents in Desai et al.'s framework, we model commitments as temporal modal operators, which in principle enable us to develop dedicated formal verification techniques that prove safety and reliability for multi-agents based web services to enter into a specific contract. It is worthwhile mentioning that classical web services as in WS-BPEL and WS-CDL cease their interactions once the demanded result is delivered, which is known as *short-lived interactions* [95]. In our modeling, the episode is entirely different as it is the beginning of the life cycle of commitments (contracts), thanks to commitment (contract) actions, which enable web services enacted and implemented by software agents to flexibly modify their established contracts.

To supply a desired outcome and enact the contracts, multi-agents based web services inter-interactions are governed and regulated by the rules of commitment (contract) protocols. As we expected, business protocols can be published to a public repository, reused, composed and aggregated. Derived from $\text{CTL}^{cc,\alpha}$, Table 16 formally specifies contract protocols in a declarative manner. In our BNF grammar, the symbols "|", "$\underline{,}$" and "::=" are used as usual to discriminate choices, enforce certain order sequence, and define non-terminal variables. This grammar consists in a group of syntactical rules: 1) the protocol

Table 16: Formal specification language of composing contract protocols

Protocol ::=	**protocol** PName$\Big\{$**role**$\langle$RNames$\rangle$
	CTL$^{cc,\alpha}$ **Formulae**$\langle$FNames$\rangle$ **message**$\langle$MNames$\rangle\Big\}$ |
	protocol PName$\Big\{$**role**$\langle$RNames$\rangle$
	CTL$^{cc,\alpha}$ **Formulae**$\langle$FNames$\rangle$ **message**$\langle$MNames$\rangle\Big\}_2$
	protocol PName$\Big\{$**role**$\langle$RNames$\rangle$
	CTL$^{cc,\alpha}$ **Formulae**$\langle$FNames$\rangle$ **message**$\langle$MNames$\rangle\Big\}$
PName ::=	CTL$^{cc,\alpha}$ atprop | CTL$^{cc,\alpha}$ atprop - CTL$^{cc,\alpha}$ atprop
RNames ::=	Role | Role, Role | Role **replace** Role
FNames ::=	Formula | Formula, Formula
MNames ::=	Message | Message, Message
Message ::=	Sender$\rightarrow$Receiver: MsgName **means**$\{$Meaning$\}$
Sender ::=	Role
Receiver ::=	Role
Meaning ::=	$\xi CC(Dt, Ct, Ant, Csq)$ | Act
ξ ::=	$S \mid W$
Dt ::=	Role
Ct ::=	Role
Ant ::=	Formula
Csq ::=	Formula
Act ::=	$Fu\xi(Dt, \xi CC(Dt, Ct, Ant, Csq))$ | $Ca\xi(Dt, \xi CC(Dt, Ct, Ant, Csq))$ | $Re\xi(Ct, \xi CC(Dt, Ct, Ant, Csq))$ | $As\xi(Ct, Ct, \xi CC(Dt, Ct, Ant, Csq))$ | $De\xi(Dt, Dt, \xi CC(Dt, Ct, Ant, Csq))$ | CTL$^{cc,\alpha}$ atomic propositions

rule (Protocol), which defines the whole protocol by calling other rules with specific ordering; 2) the protocol name rule (PName), which gives the name of protocol as an atomic proposition in CTL$^{cc,\alpha}$; 3) the rule of role names (RNames); 4) the rule of CTL$^{cc,\alpha}$ formula names (FNames); 5) the rule of message names (MNames); 6) the message rule (Message), which specifies the sender, receiver, name, and meaning of the message. We assume that messages exchanged among multi-agents based web services do not get lost and their orders are preserving; 7) the meaning rule (Meaning), which defines the pre-agreed meanings of exchanged messages using contracts and their actions such that antecedents (in the Ant rule) and consequents (in the Csq rule) of contracts are defined as CTL$^{cc,\alpha}$ formulae; and 8) the action rule (Act), which lists the allowable contract actions along with application's terminology of protocol as an atomic proposition in CTL$^{cc,\alpha}$.

Our specification language produces not only simple contract protocols, but also composite contract ones. A simple contract protocol is the one that has a single specific purpose,

such as payment, shipping, and ordering, while a composite contract protocol plausibly aggregates two or more existing simple contract protocols. This composition process is enabled in our language by repeating the syntactic protocol rule two or more times in the form of nested operation. For example, a new contract called ordering-payment protocol can be produced using the seconde choice in the protocol rule one time to compose the ordering protocol and the payment protocol in order to handle the customer's request: ordering service and payment service. This *complex* request cannot be performed by a simple contract protocol. Specifically, our composition operations and rules are as follows:

1. The *concatenation* operation in the PName rule is introduced to define the campsite protocol name.
2. The *replace* operation in the syntactical RName rule is urgently introduced to unify the name of the played roles.
3. The *sequence ordering* operation is mainly related to order the sequence of exchanged messages and their meanings. This ordering operation can implement the same function as event order axiom added between messages [36] and regulative operations (e.g., before, cause, and premise) added between active commitments [9].

To capture the real *capabilities* of our specification language in formally specifying composite contract protocols, we consider the following examples. The ordering protocol introduced in [36] can be formally specified using the introduced specification language as a simple contract protocol (see Table 17). The formal specification of the ordering protocol starts with giving the protocol name (ordering) and listing the name of participating roles (buyer and seller). By playing a role in the protocol, agents agree about the meaning of messages, and conditions of contracts. It then defines $\text{CTL}^{cc,\alpha}$ formulae. In this protocol, they are atomic propositions representing (*Quote(Presented), Price(Accepted)*, and *Goods(Accepted)*) and $\text{CTL}^{cc,\alpha}$ formulae representing (*EF Goods(Delivered)* and *EF Payment(Sent)*). These atomic propositions and formulae represent the content of the message (*requestQuoteMsg*) along with antecedents and consequences of the active contracts. These contracts define the meaning of the messages (*offerMsg* and *acceptGoodsMsg*) exchanged between buyer and seller. It is obvious that the ordering protocol composes the buyer agent-based web service and the seller agent-based web service and at the same time regulates their interactions to inter-operably work together in a harmony way. Therefore, the composition process is performed at the protocol level. As mentioned in Chapter 3, what is a duty for one party is normally a benefit for the other. The benefit of the buyer

Table 17: Formal specification of the ordering protocol

protocol	ordering {
role	Buyer, Seller
Formulae	Quote(Presented),Price(Accepted), EF Goods(Delivered), Goods (Accepted), EF Payment(Sent)
message	
Buyer→Seller:	requestQuoteMsg means $\{Request(Buyer, Seller, Quote(Presented))\}$
Seller→Buyer:	offerMsg means $\{SCC(Seller, Buyer, Price(Accepted), EF\ Goods(Delivered))\}$
Buyer→Seller:	acceptGoodsMsg means $\{SCC(Buyer, Seller, Goods(Accepted), EF\ Payment(Sent))\}$
}	

in the contract $SCC(Seller, Buyer, Price(Accepted), EF\ Goods(Delivered))$ is the delivery of the requested goods. Also, the seller's benefit in the contract $SCC(Buyer, Seller, Goods(Accepted), EF\ Payment(Sent))$ is the receiving of the payment if the delivered goods are accepted.

In a similar way, the payment protocol can be formally specified as a simple contract protocol with respect to our specification language (see Table 18). When the accepted

Table 18: Formal specification of the payment protocol

protocol	payment {
role	Payer, Payee, Bank
Formulae	Price(Accepted), EF Goods(Delivered),Goods(Accepted), EF Payment(Sent),EF Receipt(Sent)
message	
Payee→Payer:	deliverGoodsMsg means $\{FuS(Payee, SCC(Payee, Payer, Price(Accepted), EF\ Goods(Delivered))\}$
Payer→Payee:	delegateMsg means $\{DeS(Payer, Bank, SCC(Payer, Payee, Goods(Accepted), EF\ Payment(Sent))\}$
Bank→Payee:	payMsg means $\{FuS(Bank, SCC(Bank, Payee, Goods(Accepted), EF\ Payment(Sent))\}$
Payee→Payer:	receiptMsg means $\{Inform(Payee, Payer, EF\ Receipt(Sent))\}$
}	

price is sent by the payer agent-based web service, the payee agent-based web service can fulfill its contract $SCC(Payee, Payer, Price(Accepted), EF\ Goods(Delivered))$ by delivering the requested goods. The payer delegates its contract to the bank to send the agreed payment to the payee on its behalf if the delivered goods are accepted. When the payment is sent by the bank and then received by the payee, it should inform the payer by

sending the receipt.

By applying our composition operations and rules on the simple ordering and payment protocols specified above, we get a formal specification of the new composite contract protocol (see Table 19). Specifically, we used the concatenation operation to concatenate the 'ordering' and 'payment' atomic propositions together in order to define the name of the new composite protocol, which is the 'ordering-payment'. Then, we used the replace

Table 19: Formal specification of the ordering-payment protocol

protocol	ordering-payment {
role	Buyer replace Payer,
	Seller replace Payee , Bank
Formulae	Quote(Presented),Price(Accepted),EF Goods(Delivered),Goods (Accepted), EF Payment(Sent), EF Receipt(Sent)
message	
Buyer→Seller:	requestQuoteMsg means $\{Request(Buyer, Seller, Quote(Presented))\}$
Seller→Buyer:	offerMsg means $\{SCC(Seller, Buyer, Price(Accepted), EF\ Goods(Delivered))\}$
Buyer→Seller:	acceptGoodsMsg means $\{SCC(Buyer, Seller, Goods(Accepted), EF\ Payment(Sent))\}$,
Seller→Buyer:	deliverGoodsMsg means $\{FuS(Seller, SCC(Seller, Buyer, Price(Accepted), EF\ Goods(Delivered))\}$
Buyer→Seller:	delegateMsg means $\{DeS(Buyer, Bank, SCC(Buyer, Seller, Goods(Accepted), EF\ Payment(Sent))\}$
Bank→Seller:	payMsg means $\{FuS(Bank, SCC(Bank, Seller,\ Goods(Accepted), EF\ Payment(Sent))\}$
Seller→Buyer:	receiptMsg means $\{Inform(Seller, Buyer, EF\ Receipt(Sent))\}$
}	

operation to replace the payer and payee roles in the payment protocol (see Table 17) with the buyer and seller roles in the ordering protocol (see Table 18). In the formulae rule, we used the sequence ordering operation to make the ordering protocol's formulae in the front of the payment protocol's formulae and then removed the redundant ones. Finally, in the message rule, we applied the sequence ordering operation to make the ordering protocol's messages before the payment protocol's messages. The aforementioned sequence ordering can be changed to make the formal specification of the payment protocol preceding the formal specification of the ordering protocol. The resulting protocol will still work well in which all active conditional contracts should be resolved at the termination of the protocol. This result is valid as commitment (contract) protocols are flexible (see, for example, [23, 54]). Moreover, the syntax of delegation action includes three agents (bank, buyer, and

seller). Two of these agents (buyer and seller) are in the same protocol and other agent (bank) is in different protocol.

6.4.4 Experimental results of the ordering protocol

To model check the ordering service regulated by the ordering protocol, we used our modeling methodology and our model $M = (S, T, \{\sim_{i \to j} \mid (i, j) \in \mathcal{A}^2\}, \{R_{c_i} | \, i \in \mathcal{A}\}, I, \mathcal{V})$ with respect to the formal ordering protocol specification in Table 17 to generate the formal protocol model. Then, we used our automated ISPL+ template generated by MCMAS+ to encode the protocol model wherein the protocol itself is modeled as the environment agent. The encoded ISPL+ model of the seller and buyer multi-agent based web services is as follows:

```
Agent Seller
  Vars:
     s: {s0,s1,s2,s3};
     x:{x0,x1};
     z:{z0,z1};
  end Vars
Actions = {s_offer,s_com1,s_null};
Protocol:
   s=s1 : {s_offer}; s=s2 : {s_com1};
Other: {s_null};
end Protocol
Evolution:
s=s1 if s=s0 and Env.Action=p_requestQuote and Buyer.Action=
b_requestQuote;
s=s2 and x=x0 if s=s1 and Action=s_offer and Env.Action=p_offer;
s=s3 and x=x0 and z=z0 if s=s2 and Buyer.Action=
b_acceptQuote and Env.Action=p_acceptQuote;
s=s0 and x=x0 and z=z0 if s=s3 and Env.Action=p_null;
end Evolution
end Agent
Agent Buyer
    Vars:
     b: {b0,b1,b2,b3};
       x:{x0,x1};
        y:{y0,y1};
    end Vars
    Actions = {b_requestQuote,b_acceptQuote,b_null};
```

```
    Protocol:
     b=b0 : {b_requestQuote};
        b=b3 : {b_acceptQuote};
    Other: {b_null};
    end Protocol
Evolution:
b=b1 if b=b0 and Action=b_requestQuote and Env.Action=p_requestQuote;
b=b2 and x=x1 and y=y1 if b=b1 and Seller.Action=s_offer and
Env.Action=p_offer;
b=b3 and x=x0 and y=y1 if b=b2 and Env.Action=p_com1 and
Seller.Action=s_com1;
b=b0 and x=x0 and y=y1 if b=b3 and Env.Action=p_null;
end Evolution
end Agent
```

Having encoded the protocol model, we need to validate it. The validation process supported by our tool comprises in running the system interactively to check that it functions as intended. Validating the composition of web services at design time is also acknowledged in [107]. However, the authors [107] used reachability analysis techniques to check the connectivity between single services being composed w.r.t. "input/output data definition, QoS metrics and values" where composite services are modeled using Petri Nets. In our validation process, the designer can select the required transition and then check the reachable states so as to compare them with its design model. If there is an error, the designer can edit the system to address the error. This process continues until the designer is assuring that the system is effectively working as intended. Because the validation process doesn't guarantee that the intended system satisfies certain temporal properties, we have to proceed toward the verification process. To prove that the encoded protocol respects two contractual commitments that constitute the core part of the ordering protocol, we specify each commitment as a CTL$^{cc,\alpha}$ specification and combine them using the conjunction operator into a single formula:

$$\varphi_1 = EF\ SCC(Seller, Buyer, Price(Accepted), EF\ Goods(Delivered)) \wedge EF\ SCC(Buyer, Seller, Goods(Accepted), EF\ Payment(Sent))$$

This formula is a reachability property. The first part checks whether or not there exists a possibility for the seller to contractually and strongly commit to deliver the requested goods to the buyer if the buyer accepts the price. The second part can read in the same way. This formula is encoded directly in the *Formulae* section of the ISPL+ model. Using MCMAS+, we found that this formula is true.

At this point, we have to go forward to evaluate the computational performance and scalability of the developed MCMAS+ tool with respect to the ordering protocol. We achieved this goal by using two different interaction techniques and using different properties. These techniques are capable of producing extremely complex interaction models. In the first interaction technique, each agent-based web service is able to interact with all other multi-agent based web services participating in the protocol. To implement this technique, we defined our testing properties in a parametric form in which every seller interacts with every buyer, and vice versa. We call these properties *full system properties*.

$$\bigwedge_{i=1}^{m} EF\big(\bigwedge_{j=1}^{n} SCC(Seller_i, Buyer_j, Price(Accepted), EF\ Goods(Delivered))\big)$$

$$\bigwedge_{i=1}^{m} EF\big(\bigwedge_{j=1}^{n} SCC(Buyer_i, Seller_j, Goods(Accepted), EF\ Payment(Sent))\big)$$

Here, m and n refer respectively to the number of sellers and buyers. In Table 20, we solely reported five experimental results of verifying reliability of the ordering protocol against the full system properties with respect to the first technique. In the table, the number of reachable states (#States), the execution time in seconds (Time(Sec)), and the memory in use in megabytes (Memory(MB)) are as function of the number of agents (#Agents). In the last experiment, we have 10 sellers, 10 buyers and 10 protocol instances along with 200 temporal formulae produced from the above parametric formulae in which each formula is connected with the others by the conjunction operator.

Table 20: Verification results of the ordering protocol w.r.t. the first technique

#Agents	**#States**	**Time(Sec)**	**Memory(MB)**
3	5	0.14	6
6	25	0.05	6
12	625	0.34	7
24	3.9e+05	5.76	10
30	9.76562e+06	13.39	16

In the second interaction technique, a seller agent-based web service is paired with a buyer agent-based web service and all these pairs evolve in a parallel way. To implement this technique, our testing properties are defined as follows:

$$\bigwedge_{i=1}^{m} EF SCC(Seller_i, Buyer_i, Price(Accepted), EF\ Goods(Delivered))$$

$$\bigwedge_{i=1}^{m} EF\ SCC(Buyer_i, Seller_i, Goods(Accepted), EF\ Payment(Sent))$$

We call these properties *one-to-one properties*. In Table 21, we reported the verification results of the order protocol against the one-to-one properties with respect to the second interaction technique. In the last experiment, we have 10 sellers, 10 buyers and 10 protocol instances along with 20 temporal formulae.

Table 21: Verification results of the ordering protocol w.r.t. the second technique

#Agents	#States	Time(Sec)	Memory(MB)
3	5	0.14	6
6	25	0.14	6
12	625	0.22	7
24	3.9e+05	4.25	9
30	9.7656e+07	10.12	10

Since the only difference between the two interaction techniques is in the parametric form of the checked formulae, then the size of the resulting model is equal in terms of the state space. However, the first interaction technique needs more time than the second interaction technique, especially when the model is going larger. This is because the verification tool demands more time to consider the required interactions among the parallelized agents.

6.4.5 Experimental results of the payment protocol

To model check the payment service regulated by the payment protocol, we used our modeling methodology and our model $M = (S, T, \{\sim_{i\to j} \mid (i,j) \in \mathcal{A}^2\}, \{R_{c_i} \mid i \in \mathcal{A}\}, I, \mathcal{V})$ with respect to the formal payment protocol specification in Table 18 to generate the formal protocol model. Then, we used our automated ISPL+ template generated by MCMAS+ to encode the protocol model wherein the protocol itself is modeled as the environment agent. Moreover, our testing properties are as follows:

$$\varphi_2 = EF(FuS(Payee, SCC(Payee, Payer, Price(Accepted), EF\ Goods(Delivered)))$$
$$\varphi_3 = EF(DeS(Payer, Bank, SCC(Payer, Payee, Goods(Accepted),$$
$$EF\ Payment(Sent))))$$
$$\varphi_4 = EF(FuS(Bank, SCC(Bank, Payee, Goods(Accepted),\ EF\ Payment(Sent))))$$

The property φ_2 states that there exists a path such that in its future the payee fulfills her strong commitment with the payer by delivering the requested goods. This property appears simple, but it explicitly contributes well in checking the correctness of the protocol. Specifically, if the property is satisfied (i.e., the fulfillment of the commitment is achieved), then

the commitment $SCC(Payee, Payer, Price(Accepted), EF\ Goods(Delivered))$ should first hold and the commitment cannot canceled, released, assigned and delegated in the future with respect to our postulates (see Section 3.6.2 in Chapter 3). The property φ_3 expresses that there exists a path such that in its future the payer delegates her strong commitment to the bank to send the payment to the payee. The satisfaction of this property means that the interaction composition of the multi-agents based web services (payee, payer and bank) is done successfully and formally according to the semantics of delegation action. This semantics ensures that the strong commitment $SCC(Payer, Payee, Goods(Accepted), EF\ Payment(Sent))$ between the payer and payee should terminate first in order to establish a new commitment $SCC(Bank, Payee, Goods(Accepted), EF\ \ Payment(Sent))$ between the bank and payee. The property φ_4 can be read and discussed as φ_2.

By using MCMAS+, we found that these properties are satisfied; so the above discussions are valid. The verification results of the payment protocol are reported in Tables 22 and 23. As we did above, from the second experiment, φ_2, φ_3 and φ_4 are defined in a parametric form. In the last experiment in Tables 22 and 23, we tested the correctness of the

Table 22: Verification results of the payment protocol w.r.t. the first technique

#Agents	**#States**	**Time(Sec)**	**Memory(MB)**
4	10	0.102	6
8	100	0.317	7
12	1000	1.479	8
16	10000	26.525	14
20	1e+05	32.529	10
24	1e+06	38.099	18
28	1e+07	53.884	15

Table 23: Verification results of the payment protocol w.r.t. the second technique

#Agents	**#States**	**Time(Sec)**	**Memory(MB)**
4	10	0.102	6
8	100	0.217	7
12	1000	1.117	8
16	10000	12.273	13
20	1e+05	15.423	10
24	1e+06	17.673	10
28	1e+07	18.541	12

payment protocol against respectively 21 and 141 temporal formulae in which each formula

is connected with the others by the conjunction operator. Because there are several interactions among multi-agent based web services that impose MCMAS+ to maintain more social states until the established interactions terminate, we can readily observe from Tables 22 and 23 that the first interaction technique requires more execution time and memory in use than the second interaction technique. For the readability purpose, Figure 21 graphically compares between the two techniques in terms of execution time. Furthermore, the experimental results reported in the last two columns in Tables 22 and 23 typically conform our proved theoretical result, which is polynomial space (or PSPACE-completeness).

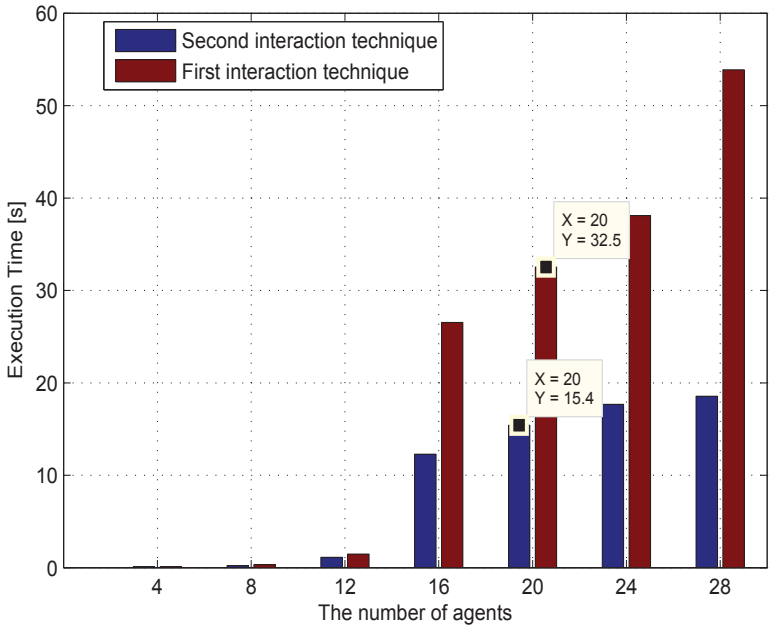

Figure 21: Comparison results between the proposed interleaved interaction techniques

Figure 22 illustrates part of the generated labeled transition system of the payment protocol model using the extended and developed graphical user interface. Displaying the protocol model enables the designers and prospective users to track the model through the validation process, especially when a counterexample is generated. Ultimately, the outcome of the validation process is to insure that the right regulation process was built (i.e., the design fits the requirements). So, the validation process is a useful step prior to implement the complex interaction models of autonomous and heterogeneous agents regulated by contract protocols.

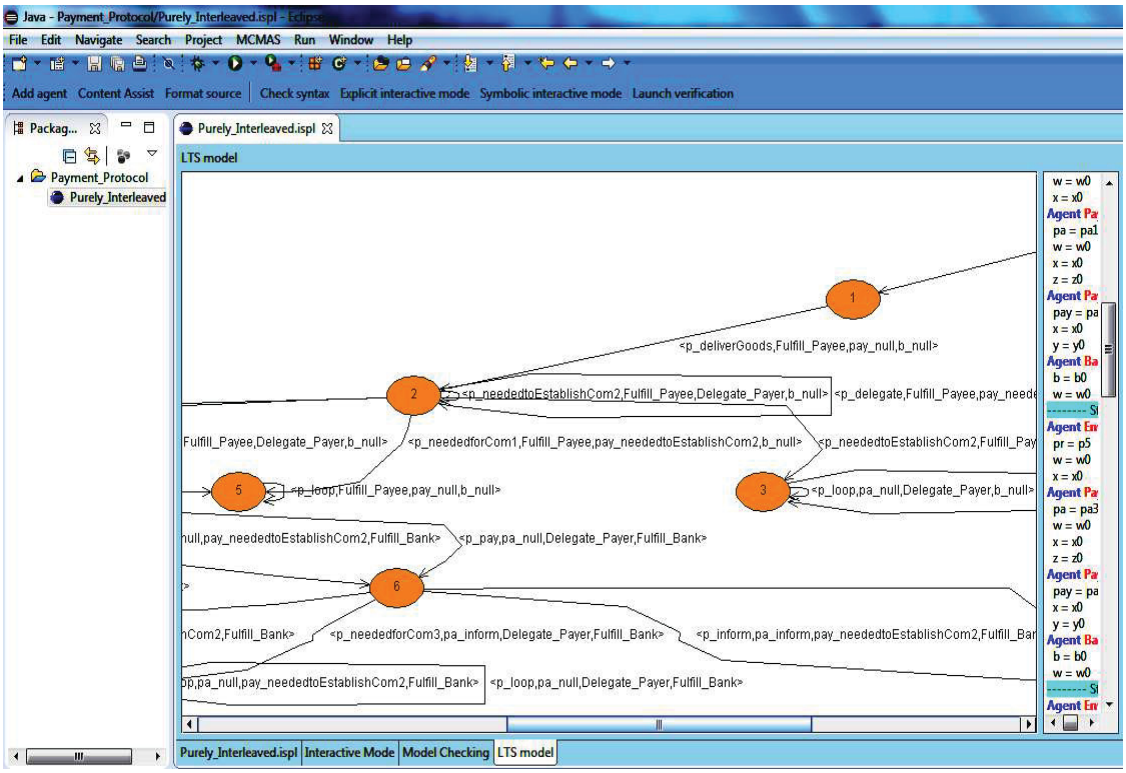

Figure 22: The generated labeled transition system of the payment protocol model

6.4.6 Experimental results of composite protocols

To model check the ordering and payment services regulated by the composite ordering-payment protocol, we used our modeling methodology and our model $M = (S, T, \{\sim_{i\rightarrow j} \mid (i, j) \in \mathcal{A}^2\}, \{R_{c_i} \mid i \in \mathcal{A}\}, I, \mathcal{V})$ with respect to the formal ordering-payment protocol specification in Table 19 to generate the formal protocol model. We then used our automated ISPL+ template generated by MCMAS+ to encode the protocol model in order to automatically verify its reliability and correctness. Hereafter, CTL$^{cc,\alpha}$ are utilized for two purposes. First, it can formally express the properties (like the above φ_1, φ_2, φ_3, and φ_4) that participating multi-agent based web services are required to fulfill. Such properties indeed formalize the assumptions on the simple ordering and payment protocols made by the software developers at the development time while designing the choreography of these protocols. We call these properties *assumed properties*. Second, CTL$^{cc,\alpha}$ can be employed to state reasonable and intuitive properties that the choreography of the composite ordering-payment protocol should satisfy, assuming that its simple ordering and payment protocols

function as specified. The properties, which qualify the behavior of the composite protocol should guarantee, are called *guaranteed properties*. In fact, the assume-guarantee reasoning is a well-known verification technique, which supports for example "divide and conquer" compositional reasoning [32].

Having automatically checked the correctness of the ordering and payment protocols against the assumed properties and found that these protocols successfully work as they are specified, we focus thereafter on guaranteed properties. Specifically, we classified guaranteed properties into *safety, liveness, response* and *guarantee* properties.

1. Safety property. The bad situation is that the buyer delegates her strong commitment to the bank to pay to the seller on her behalf, but the seller never sends the receipt back:

   $AG\neg(DeS(Buyer, Bank, SCC(Buyer, Seller, Goods(Accepted),$
   $EF\ Payment(Sent)) \wedge \neg EF\ Receipt(Sent)))$

2. Liveness property. The good situation is that along all future paths when the seller delivers the requested goods, there is a path where the buyer will contractually and strongly commit to send the agreed payment:

   $AF(Goods(Delivered) \wedge EF\ SCC(Buyer, Seller, Goods(Accepted),$
   $EF\ Payment(Sent)))$

   Notice that the safety and liveness properties forge cross-fertilizations among the ordering and payment protocols in which the delegated commitment is appeared in the ordering protocol and the new commitment and payment are appeared in the payment protocol.

3. Response property. For example, when the seller fulfills her commitment by delivering the requested goods, her commitment will be terminated:

   $AG\big(FuS(Seller, SCC(Seller, Buyer, Price(Accepted), EF\ Goods(Delivered)))$
   $\rightarrow AF\neg SCC(Seller, Buyer, Price(Accepted), EF\ Goods(Delivered))\big)$

4. Guarantee property. For example, at least one state along all paths satisfies that the seller will contractually commit to the buyer to deliver the goods until the buyer has sent a request:

   $AFA(\neg request\ U\ request \wedge SCC(Seller, Buyer, Price(Accepted),$
   $EF\ Goods(Delivered)))$

To continue checking the computational performance and scalability of the developed model checker, Table 24 reports the verification results of the composite ordering-payment protocol with respect to the second interaction technique and with different 4 properties.

Table 24: Verification results of the ordering-payment protocol w.r.t. the second technique

#Agents	#States	Time(sec)	Memory(MB)
4	11	0.062	6
8	121	0.280	7
12	1331	1.684	8
16	14641	2.605	9
20	161051	9.952	10
24	1.77156e+06	13.634	11
28	1.94872e+07	20.108	16
32	2.14359e+08	91.050	13
36	2.35795e+09	127.272	16
40	2.59374e+10	278.538	23

The obtained results showcase that when the number of agents increases, the number of reachable states (i.e., state-space) increases exponentially, while the memory in use increases polynomially, which experimentally confirms the theoretical space complexity result, although the safety and response properties require to check all states along all paths in the protocol. Moreover, in the last experiment, we tested the satisfaction of 40 temporal formulae within an ISPL+ model having 2.59374e+10 state space and aggregated from 40 business models of multi-agent based web services in 278.538 seconds with 23 megabytes.

Figure 23 graphically illustrates the polynomial relationship between the memory in use and the number of agents. In the figure, the bars for 32 and 36 agents are not higher than the bar of 28 agents. It results from OBDDs encoding, which may change from one model to another based on some internal optimization techniques implemented in the original version of MCMAS. However, it is easy to analytically and mathematically compute this polynomial relation considering the whole data: 4 to 40 agents. Specifically, the relation is of order 3 and can be computed as: $y = 0.0006x^3 - 0.0274x^2 + 0.6618x + 3.4$, where y is the memory usage and x is the number of employed agents. Thus, when $x = 4$ and $x = 100$, then $y = 5.6472$ and $y = 395.58$, respectively. Finally, since our tool is the first one that can model-check conditional commitment actions in the literature, we did not have a chance to compare our results.

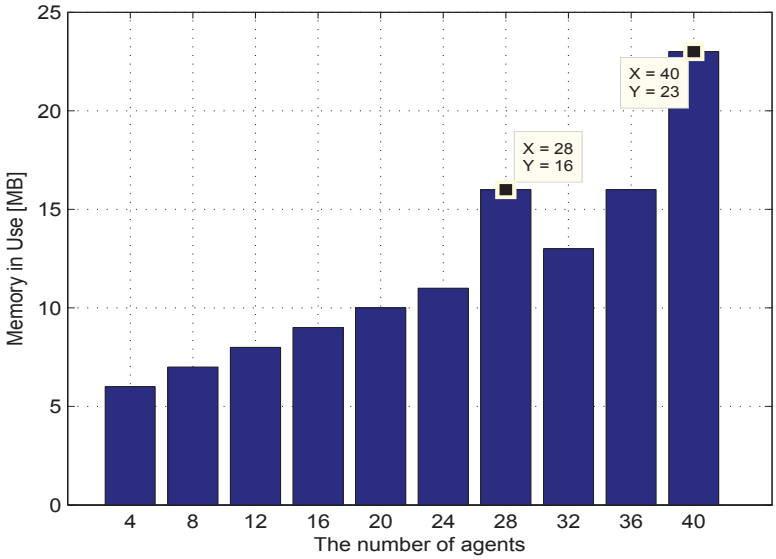

Figure 23: The memory in use vs. the number of agents

6.4.7 Reviewing related work of web service composition

In this section, we discuss and review related work. Specifically, we group current approaches to model web service compositions into conventional software engineering approaches, commitment-based approaches, and multi-agent approaches. The former group strongly advocates traditional techniques, such as AI planning, process algebra, and Petri nets. The idea of composing services employed in this group is intimately related to workflow models that control logic required to coordinate data over partner services using different constructs (e.g., sequence and parallel). Thus, the workflow technique becomes the core element in the standard service composition languages such as BPEL4WS [1] and WSCDL [13]. For example, BPEL4WS divides business processes into executable and abstract processes. The former process models the behaviour of participating services in the form of a private workflow that controls the flow of data through synchronous and asynchronous invocation methods. The latter processes publicly specify the sequence of message exchanges among parties. Such techniques exploited in BPEL4WS and WSCDL have been recently criticized in the commitment-based approaches [27], as they mostly gear for low-level details (i.e., how services exchange messages) to achieve interoperability (the interaction is deadlock free), rather than offering high-level abstractions (i.e., what the messages mean in the real world).

The suggestion of using social commitments into the context of SOA was pioneered by Singh and Huhns in their book [83]. They specifically improved current SOA architectures, interpreting services narrowly as computational objects to use invocation methods, by giving primacy to the business meanings of service engagements. Computationally, agents perform service engagements (e.g., payment service) by creating and manipulating commitments to one another using a set of commitment actions. In [95], they pointed out that BPEL4WS don't consider the complexity of domain level logic and the size of the service models when augmenting tasks and their compensations to consider dynamic changes. The authors then enhanced web service descriptions with commitments and their actions. However, they only concentrated on the formulation of commitments and their actions in the XML document, which in turn leaves the formulation of the message syntax and semantics open. Xing and Singh [102] proposed a set of commitment patterns inspired by object-design patterns to model agent interactions. Commitments are simply modeled as abstract data types where the commitment consequence is defined as a predicate, and commitment actions are modeled simply as propositions. The authors then exploited a statechart to specify the behavior model of each agent. The authors in [103] developed an algorithm to transform the statechart of agent's behavior model into CTL model. Their composition process is based on using the concept of object designs, called "Merge" to compose "multiple statecharts into a statechart" to establish agents' behavior model. However, the concepts of conditional commitments (i.e., commitments that become active provided some condition holds), and commitment-based protocols are not considered. The main limitation of the approaches [82, 95, 102, 103] is the lack of formal semantics of commitments and checking the compliance of committing agent behaviors with specifications automatically.

The authors in [92] tried to solve the problem of verifying agent interactions in the context of cross-organizational business processes. They exploited the NuSMV model checker to check whether or not an operational model correctly supports business patterns formalized as CTL formulae. However, the way of formally composing the interactions among business processes and the internal design of agents implementing business processes are omitted. Furthermore, the formal semantics of commitments and their actions are missing.

Multi-agent approaches focus on regulating the business interactions among multi-agent based web services within the composition process using contracts, conversation protocols, and business protocols. Among them, Lomuscio et al. [65] modeled all possible behaviors of multiagent-based services and the correct behaviors for every agent regulated by binding electronic contracts as BPEL4WS specifications. Such specifications are

transformed into ISPL (the input language of MCMAS [64]) to verify desirable properties needed for service compositions and formalized by temporal-epistemic logic, extending CTL with the knowledge modality and local atomic propositions. However, the direct interaction between the contract's parties are not captured and the formalization of the contract itself is abstracted away. Fu et al. [58] modeled services as agent peers interacting via asynchronous messages in XML format. They presented a compiler translating the conversation protocol to PROMELA and showed that properties of conversation protocols, expressed in LTL, can be automatically verified using the SPIN model checker. However, the proposed model concentrates on control flows and data manipulation semantics, which restricts the flexibility and modularity of protocols. The authors in [36] developed a language, called OWL-P, which incorporates the standard web ontology language (OWL) and a set of primitives to specify business protocols such as roles, messages and their social meanings exchanged between them. Each business process reflects a composition of a set of business protocols w.r.t. some properties expressed in pi-calculus. However, OWL-P waives formal semantics for social commitments. Also, they do not consider how to verify composite protocols.

Chapter 7

Conclusion and Future Work

This chapter covers:

- A summary of the obtained results.
- A presentation of open issues.
- A sketch of possible extension of this work.

7.1 Conclusion

Specifying and verifying interactions among autonomous and possibly heterogeneous software intelligent agents, modeled in terms of social commitments and their actions from high-level abstractions, provide the quintessential basis for constructing effective open MASs. In our thinking, effective MASs are those that successfully achieve error-free design, balance among expressive power, verification efficiency and feasibility aspects. The companion contribution of this dissertation lies in presenting a novel operational approach which: 1) formally defines a new computationally grounded semantics for ACL messages in terms of two types of conditional commitments and their duplex and triplex actions; and 2) develops a new symbolic model checking algorithm dedicated to CTL$^{cc,\alpha}$, an extension of CTL with temporal modalities to represent and reason about strong and weak conditional commitments and their associated actions (fulfill, cancel, release, assign, and delegate). Moreover, the proposed approach has five salient features distinguishing it from the literature of agent communication. First, the specificity of weak commitment is its compatibility with the literature and of strong commitment is in advancing the literature towards abstractly modeling real and concrete applications that weak commitments cannot model. Second, the proposed semantics of fulfillment modalities successfully remedy the

spurious paradox that plaques current semantic models. Third, our specification commitment language CTL$^{cc,\alpha}$ supports a set of valid properties and also supports all reasoning rules and action postulates commonly accepted in the literature and agents are expected to respect them when they communicate. Four, the developed algorithm is fully implemented on top of the model checker MCMAS producing thus a new one called MCMAS+. The soundness and termination of the developed algorithm are proved in a theoretical manner. We also extended the MCMAS's input language ISPL and graphical user interface to support new modalities and shared and unshared variables and to render the labeled transition systems of models to help designers edit, design, track, validate and view models. Five, the model checking of the proposed CTL$^{cc,\alpha}$ is efficient as its time complexity is P-complete and its space complexity is PSPACE-complete, which means polynomial in both time and space.

In terms of the feasibility aspect, we successfully applied our approach in three different application domains (business interaction protocols, health care processes, and web service compositions). The MAS paradigm was introduced in these applications to particularly improve the management of taking required decisions and actions and govern interactions among participants wherein the main components of these applications are represented, implemented and enacted by intelligent agents. The proposed approach improved the employed MAS paradigm by formally specifying and automatically verifying interactions among agents so that the bad behaviors can be detected and then eliminated or repaired at design time, which entails the reduction of the post-development costs and the increase of the confidence on the safety, efficiency and robustness. Specifically, we introduced a methodology to model the formal or informal specification of MASs and a library of properties as well as a formal language to specify and compose commitment (contract) based protocols derived from CTL$^{cc,\alpha}$. We then used the NetBill protocol, the process of breast cancer diagnosis and treatment, the simple contract protocol regulating interactions of the ordering service, the simple contract protocol regulating interactions of the payment service, and the composite contract protocol regulating interactions of the ordering and payment services as case studies in order to experimentally evaluate the computational performance and scalability of our model checker MCMAS+. The automated ISPL+ template generated by MCMAS+ is used to encode these case studies formalized using our model in a systematic way. The obtained experimental results strongly conform our theoretical finding (the polynomial space) and certify the computational performance and scalability of MCMAS+ by considering very large and different case studies having approximately

(10^{16} states in Table 12, 10^{8} states in Table 15 and 10^{10} states in Table 24) with respect to two interleaved interaction techniques, thanks to the OBDDs-based symbolic encodings employed in MCMAS+. Moreover, we showed how to analytically and mathematically compute the polynomial relation to compute the expected memory consumption given the number of interacting agents (e.g., 100 agents). To this end, our approach is clearly not exhaustive, but helps designers validate and automatically check the compliance of the behaviors of interacting agents, MASs specifications and protocol specifications with temporal properties wherein social conditional commitments and their actions modalities form the core part of these properties. When comparing our approach to other available approaches in the literature, we found that it considerably needs low execution time and memory usage to successfully perform the verification task, and considers more number of interacting intelligent agents (45 agents).

7.2 Future work

Despite we successfully applied our specification commitment language CTL$^{cc,\alpha}$ and its symbolic model checker MCMAS+ along with its input language ISPL+ to three different application domains, some issues still need to be tackled. The following open issues are not considered in the dissertation:

– A considerably large class of MASs employed in real-time environments requires the possibility to express time-critical properties. Such properties indeed express the occurrences of events at time instants or within time intervals and play an essential role in verifying the correctness of the specifications of MASs. The current version of CTL$^{cc,\alpha}$ is unable to express the quantitative property stating that a seller has a commitment with a buyer to deliver the requested goods after 3 business days from the time of receiving the agreed payment. We developed a system of temporal logic called RTCTL^{cc}, an extension of CTL with interval bound until modalities and conditional commitments and their fulfillment modalities [45]. This logic combines qualitative temporal aspects together with real-time constraints in order to permit reasoning about qualitative and quantitative requirements. However, encoding unit transition steps in the RTCTL^{cc} model leads to quite costly extra verification work. Moreover, the timing requirements of other commitment actions (i.e., actions that require time to be completed) and the problem of model checking RTCTL^{cc} are yet to be investigated.

We also plan to:

- Prove theoretically the completeness of an axiomatic system (say Γ) for $\text{CTL}^{cc,\alpha}$ in order to complement our practical work on model checking. The completeness problem states that any true statement φ in Γ can be established by proof steps in the logic's calculus, formally: $\Gamma \models \varphi$ implies $\Gamma \vdash \varphi$.
- Analyze the relationships between agent communication commitments and commitments in strategic logics such as the one studied in [2].
- Study the logical relationships between probability and conditional commitments and their actions to express uncertain communication properties, as done in the case of unconditional commitments and their fulfillments [87].
- Study the relationship between trust and conditional commitments and their actions from a logical perspective.
- Investigate the possibility of extending $\text{CTL}^{cc,\alpha}$ with first-order quantifiers to reason about conditional commitments and their actions by: 1) following the methodology introduced by Belardinelli et al. [14], which extends CTLK with first-order quantifiers; and 2) lifting the antecedents and consequences of commitments to a first-order setting in order to study how data represented as first-order formulae and maintained by interacting agents affect the evolution of commitments in the system, as done by Montali et al. [71].

  1. The advantage of Belardinelli et al.'s methodology is that it provides a systemic way to find bisimilar finite abstractions, which reduce the model checking problem to the instance on finite models.
  2. The fundamental results of Montali et al.'s approach will enable us to establish the decidability of the verification of temporal properties under the condition of state-boundedness by using a finite number of symbolic terms to abstractly represent real, first-order data.

  We expect this extension will improve the commitment contents with predicates, which capture data and domain variables in a natural way.

Bibliography

[1] Web services business process execution language version 2.0 (WS-BPEL 2.0), https://www.oasis-open.org/committees/download.php/23964/wsbpel-v2.0-primer.htm_Toc166509724

[2] Ågotnes, T., Goranko, V., Jamroga, W.: Strategic commitment and release in logics for multi-agent systems (extended abstract). Tech. Rep. IfI-08-01, Clausthal University of Technology (2008)

[3] Al-Saqqar, F., Bentahar, J., Sultan, K., El-Menshawy, M.: On the interaction between knowledge and social commitments in multi-agent systems. Applied Intelligence 41(1), 235–259 (2014)

[4] Al-Saqqar, F., Bentahar, J., Sultan, K., Wan, W., Asl, E.: Model checking temporal knowledge and commitments in multi-agent systems using reduction. Simulation Modelling Practice and Theory 51, 45–68 (2015)

[5] Arkin, A., Askary, S., Fordin, S., Jekeli, W., Kawaguchi, K., Orchard, D., Pogliani, S., Riemer, K., Struble, S., Takacsi-Nagy, P., Trickovic, I., Zimek, S.: Web service choreography interface (WSCI 1.0) (August 2002), http://www.w3.org/TR/2002/NOTE-wsci-20020808

[6] Artikis, A., Pitt, J.V.: Specifying open agent systems: A survey. In: Artikis, A., Picard, G., Vercouter, L. (eds.) ESAW. LNCS, vol. 5485, pp. 29–45. Springer (2009)

[7] Austin, J.: How to do things with words. Oxford University Press: Oxford, England (1962)

[8] Baldoni, M., Baroglio, C., Marengo, E.: Behavior oriented commitment-based protocols. In: Coelho, H., Studer, R., Wooldridge, M. (eds.) ECAI. vol. 215, pp. 137–142 (2010)

[9] Baldoni, M., Baroglio, C., Marengo, E., Patti, V.: Constitutive and regulative specifications of commitment protocols: A decoupled approach. ACM Trans. on Inte. Syst. and Tech. 4(2), 22 (2013)

[10] Baldoni, M., Baroglio, C., Capuzzimati, F., Marengo, E., Patti, V.: A generalized commitment machine for 2CL protocols and its implementation. In: Baldoni, M., Dennis, L.A., Mascardi, V., Vasconcelos, W. (eds.) Proceedings of Declarative Agent Languages and Technologies X - 10th International Workshop. LNCS, vol. 7784, pp. 96–115. Springer (2013)

[11] Baldoni, M., Baroglio, C., Marengo, E., Patti, V., Capuzzimati, F.: Engineering commitment-based business protocols with the 2CL methodology. Autonomous Agents and Multi-Agent Systems 28(4), 519–557 (2014)

[12] Baldoni, M., Baroglio, C., Martelli, A., Patti, V.: Reasoning about interaction protocols for customizing web service selection and composition. Logic and Algebraic Programming 70, 53–73 (2007)

[13] Barros, A., Dumas, M., Oaks, P.: A critical overview of the web services choreography description language (WS-CDL). Business Process Trends (March 2005), http://www.bptrends.com

[14] Belardinelli, F., Lomuscio, A., Patrizi, F.: Verification of agent-based artifact systems. Artificial Intelligence Research 51, 333–376 (2014)

[15] Bentahar, J., El-Menshawy, M., Qu, H., Dssoulia, R.: Communicative commitments: Model checking and complexity analysis. Knowledge-Based Systems 35, 21–34 (2012)

[16] Bentahar, J., Meyer, J.J., Wan, W.: Model checking agent communication. In: Dastani, M., Hindriks, K., Meyer, J.J. (eds.) Specification and Verification of Multi-Agent Systems, chap. 3. Springer, first edn. (2010)

[17] Bentahar, J., Moulin, B., Meyer, J.J., Lespérance, Y.: A new logical semantics for agent communication. In: Inoue, K., Satoh, K., Toni, F. (eds.) CLIMA. LNCS, vol. 4371, pp. 151–170. Springer (2007)

[18] Bhat, G., Cleaveland, R., Groce, A.: Efficient model checking via Büchi tableau automata. In: Berry, G., Comon, H., Finkel, A. (eds.) CAV. LNCS, vol. 2102, pp. 38–52. Springer (2001)

[19] Bhat, S., Sidnal, N.S., Malashetty, R.S., Manvi, S.S.: Intelligent scheduling in health care domain. IJCSI International Journal of Computer Science Issues 8(2), 214–224 (2011)

[20] Booth, D., Haas, H., McCabe, F., Newcomer, E., Champion, M., Ferris, C., Orchard, D.: Web services architecture. W3C Working Group Note (February 2004), http://www.w3.org/TR/ws-arch/

[21] Bordini, R.H., Dastani, M., rgen Dix, J., Seghrouchni, A.E.F. (eds.): Multi-agent programming languages, platforms and applications. Springer (2005)

[22] Chakraborty, S., Gupta, S.: Medical application using multi agent system: A literature survey. Engineering Research and Applications 4(2), 528–546 (2014)

[23] Chesani, F., Mello, P., Montali, M., Torroni, P.: Representing and monitoring social commitments using the event calculus. Autonomous Agents and Multiagent Systems 27(1), 85–130 (2013)

[24] Chopra, A., Singh, M.: Nonmonotonic commitment machines. In: Dignum, F. (ed.) ACL. LNCS, vol. 2922, pp. 183–200. Springer (2004)

[25] Chopra, A., Singh, M.: Contextualizing commitment protocols. In: Nakashima, H., Wellman, M., Weiss, G., Stone, P. (eds.) AAMAS. pp. 1345–1352. ACM (2006)

[26] Chopra, A., Singh, M.: Multiagent commitment alignment. In: Proc. of the 8th Int. Joint Conf. on AAMAS. pp. 937–944. ACM Press (2009)

[27] Chopra, A., Singh, M.: Agent communication. In: Weiss, G. (ed.) Multiagent systems: A modern approach to distributed artificial intelligence, chap. 3. MIT Press, second edn. (2013)

[28] Chopra, A.K., Artikis, A., Bentahar, J., Colombetti, M., Dignum, F., Fornara, N., Jones, A.J.I., Singh, M.P., Yolum, P.: Research directions in agent communication. ACM Trans. on Inte. Syst. and Tech. 4(2), 20 (2013)

[29] Cimatti, A., Clarke, E., Giunchiglia, E., Giunchiglia, F., Pistore, M., Roveri, M., Sebastiani, R., Tacchella, A.: NuSMV: An open source tool for symbolic model checking. In: Brinksma, E., Larsen, K.G. (eds.) CAV. LNCS, vol. 2404, pp. 359–364. Springer (2002)

[30] Clarke, E., Emerson, E.: Design and synthesis of synchronization skeletons using branching time temporal logic. In: Kozen, D. (ed.) Logics of Programs. LNCS, vol. 131, pp. 52–71 (1982)

[31] Clarke, E., Emerson, E., Sistla, A.: Automatic verification of finite-state concurrent systems using temporal logic specifications. In: Proc. of the 10th ACM SIGACT-SIGPLAN Symposium on PPL. pp. 117–126. POPL'83, ACM (1986)

[32] Clarke, E., Grumberg, O., Peled, D.: Model Checking. The MIT Press, Massachusetts (1999)

[33] Clarke, E.M., Emerson, E.A., Sifakis, J.: Model checking: Algorithmic verification and debugging. Communications of the ACM 52(11), 74–84 (2009)

[34] Desai, N., Cheng, Z., Chopra, A., Singh, M.: Toward verification of commitment protocols and their compositions. In: Durfee, E.H., Yokoo, M., Huhns, M.N., Shehory, O. (eds.) AAMAS. pp. 144–146. IFAAMAS (2007)

[35] Desai, N., Chopra, A., Singh, M.: Amoeba: A methodology for modeling and evolution of cross-organizational business processes. ACM Trans. on Soft. Eng. and Metho. 19(2), 1–40 (2009)

[36] Desai, N., Mallya, A., Chopra, A., Singh, M.: Interaction protocols as design abstractions for business processes. IEEE Trans. on Soft. Eng. 31(12), 1015–1027 (2005)

[37] Desai, N., Singh, M.: A modular action description language for protocol composition. In: Proc. of the 22nd AAAI Conf. on Artificial Intelligence. pp. 962–967 (2007)

[38] Desai, N., Singh, M.: On the enactability of business protocols. In: Fox, D., Gomes, C.P. (eds.) AAAI. pp. 1126–1131. AAAI Press (2008)

[39] El-Kholy, W., Bentahar, J., El-Menshawy, M., Qu, H., Dssouli, R.: Conditional commitments: Reasoning and model checking. ACM Trans. on Soft. Eng. and Metho. 24(2), 9:1–9:49 (2014)

[40] El-Kholy, W., Bentahar, J., El-Menshawy, M., Qu, H., Dssouli, R.: Modeling and verifying choreographed multi-agent-based web service compositions regulated

by commitment protocols. Expert Systems with Applications 41(16), 7478–7494 (2014)

[41] El-Kholy, W., El-Menshawy, M., Bentahar, J., Qu, H., Dssouli, R.: Representing and reasoning about communicative conditional commitments. In: Gini, M.L., Shehory, O., Ito, T., Jonker, C.M. (eds.) Proceedings of the International Conference on AAMAS. pp. 1169–1170. IFAAMAS (2013)

[42] El-Kholy, W., El-Menshawy, M., Bentahar, J., Qu, H., Dssouli, R.: Verifying multiagent-based web service compositions regulated by commitment protocols. In: Proceedings of the International Conference on Web Services. pp. 49–56. IEEE Computer Society (2014)

[43] El-Kholy, W., El-Menshawy, M., Bentahar, J., Qu, H., Dssouli, R.: Formal specification and automatic verification of conditional commitments. IEEE Intelligent Systems 30(2), 36–44 (2015)

[44] El-Kholy, W., El-Menshawy, M., Bentahar, J., Qu, H., Dssouli, R.: SMC4AC: A new symbolic model checker for agent communication. Fundamenta Informaticae xxx, 1–42 (2015), IOS Press (Submitted)

[45] El-Kholy, W., El-Menshawy, M., Laarej, A., Bentahar, J., Al-Saqqar, F., Dssouli, R.: Real-time conditional commitment logic. In: Chen, Q., Torroni, P., Villata, S., Hsu, J.Y., Omicini, A. (eds.) Proceedings of the 18th International Conference on Principles and Practice of Multi-Agent Systems. vol. 9387, pp. 547–556 (2015)

[46] El-Menshawy, M.: Model Checking Logics of Social Commitments for Agent Communication. Ph.D. thesis, Concordia University (2012)

[47] El-Menshawy, M., Bentahar, J., Dssouli, R.: Enhancing engineering methodology for communities of web services. In: Baldoni, M., et al. (eds.) MALLOW. vol. 494. CEUR-WS.org (2009)

[48] El-Menshawy, M., Bentahar, J., Dssouli, R.: Modeling and verifying business interactions via commitments and dialogue actions. In: Jedrzejowicz, P., Nguyen, N.T., Howlett, R.J., Jain, L.C. (eds.) KES-AMSTA. LNCS, vol. 6071, pp. 11–21 (2010)

[49] El-Menshawy, M., Bentahar, J., Dssouli, R.: Verifiable semantic model for agent interactions using social commitments. In: Dastani, M., Fallah-Seghrouchni, A.E., Leite, J., Torroni, P. (eds.) LADS. LNCS, vol. 6039, pp. 128–152 (2010)

[50] El-Menshawy, M., Bentahar, J., Dssouli, R.: Model checking commitment protocols. In: Mehrotra, K.G., et al. (eds.) IEA–AIE. LNCS, vol. 6704, pp. 37–47 (2011)

[51] El-Menshawy, M., Bentahar, J., Dssouli, R.: Symbolic model checking commitment protocols using reduction. In: Omicini, A., Sardina, S., Vasconcelos, W. (eds.) DALT. LNAI, vol. 6619, pp. 185–203. Springer (2011)

[52] El-Menshawy, M., Bentahar, J., Kholy, W.E., Dssouli, R.: Reducing model checking commitments for agent communication to model checking ARCTL and GCTL*. Autonomous Agent Multi-Agent Systems 27(3), 375–418 (2013)

[53] El-Menshawy, M., Bentahar, J., Qu, H., Dssouli, R.: On the verification of social commitments and time. In: Sonenberg, L., Stone, P., Tumer, K., Yolum, P. (eds.) AAMAS. pp. 483–490. IFAAMAS (2011)

[54] El-Menshawy, M., Bentahar, J., El-Kholy, W., Dssouli, R.: Verifying conformance of multi-agent commitment-based protocols. Expert Systems with Applications 40(1), 122–138 (2013)

[55] El-Menshawy, M., Bentahar, J., El-Kholy, W., Dssouli, R.: Computational logics and verification techniques of multi-agent commitments: Survey. The Knowledge Engineering Review 30(5), 564–606 (2015), cambridge University Press

[56] Eshuis, R.: Symbolic model checking of uml activity diagrams. ACM Trans. on Soft. Eng. and Metho. 15(1), 1–38 (2006)

[57] Fagin, R., Halpern, J., Moses, Y., Vardi, M.: Reasoning about Knowledge. The MIT Press, Cambridge (1995)

[58] Fu, X., Bultan, T., Su, J.: Analysis of interacting BPEL web services. In: Feldman, S.I., Uretsky, M., Najork, M., Wills, C.E. (eds.) Proceedings of the 13th international conference on World Wide Web (WWW 2004). pp. 621–630 (2004)

[59] Gerard, S., Singh, M.: Formalizing and verifying protocol refinements. ACM Trans. on Intel. Syst. and Tech. 4(2), 21 (2013)

[60] Gupta, S., Sarkar, A., Pramanik, I., Mukherjee, B.: Implementation scheme for online medical diagnosis system using multi agent system with JADE. Scientific and Research Publications 2(6), 2250–3153 (2012)

[61] Özgür Kafali, Günay, A., Yolum, P.: Detecting and predicting privacy violations in online social networks. Distributed and Parallel Databases 32, 161–190 (2014)

[62] Kupferman, O., Vardi, M., Wolper, P.: An automata-theoretic approach to branching-time model checking. ACM 47(2), 312–360 (2000)

[63] Lomuscio, A., Pecheur, C., Raimondi, F.: Automatic verification of knowledge and time with NuSMV. In: Proc. of the 20th Int. Joint Conf. on AI. pp. 1384–1389 (2007)

[64] Lomuscio, A., Qu, H., Raimondi, F.: MCMAS: A model checker for the verification of multiagent systems. In: Bouajjani, A., Maler, O. (eds.) CAV. LNCS, vol. 5643, pp. 682–688. Springer (2009)

[65] Lomuscio, A., Qu, H., Solanki, M.: Towards verifying contract regulated service composition. Autonomous Agents and Multi-Agent Systems 24(3), 345–373 (2012)

[66] Mallya, A., Huhns, M.: Commitments among agents. IEEE Internet Computing 7(4), 90–93 (2003)

[67] Mallya, A., Singh, M.: An algebra for commitment protocols. Autonomous Agents and Multi-Agent Systems 14(2), 143–163 (2007)

[68] Mallya, A., Yolum, P., Singh, M.: Resolving commitments among autonomous agents. In: Dignum, F. (ed.) ACL. LNCS, vol. 2922, pp. 166–182. Springer (2004)

[69] Manna, Z., Pnueli, A.: The Temporal Logic of Reactive and Concurrent Systems: Specificatian. Springer (1991)

[70] Marengo, E., Baldoni, M., Baroglio, C., Chopra, A., Patti, V., Singh, M.: Commitments with regulations: Reasoning about safety and control in REGULA. In: Tumer, K., Yolum, P., Sonenberg, L., Stone, P. (eds.) AAMAS. pp. 467–474 (2011)

[71] Montali, M., Calvanese, D., Giacomo, G.D.: Verification of data-aware commitment-based multiagent systems. In: Lomuscio, A., Scerri, P., Bazzan, A., Huhns, M. (eds.) Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2014). pp. 157–164. IFAAMAS (2014)

[72] Moy, Y., Ledinot, E., Delseny, H., Wiels, V., Monate, B.: Testing or formal verification: DO-178C alternatives and industrial experience. IEEE Software 30(3), 50–57 (2013)

[73] Pecheur, C., Raimondi, F.: Symbolic model checking of logics with actions. In: Edelkamp, S., Lomuscio, A. (eds.) Model Checking and Artificial Intelligence. LNCS, vol. 4428, pp. 113–128. Springer (2007)

[74] Peltz, C.: Web services orchestration and choreography. IEEE Computer 36(10), 46–52 (2003)

[75] Penczek, W., Lomuscio, A.: Verifying epistemic properties of multiagent systems via bounded model checking. Fundamenta Informaticae 55(2), 167–185 (2003)

[76] Pham, D., Harland, J.: Temporal linear logics as a basis for flexible agent interactions. In: Durfee, E., Yokoo, M., Huhns, M., Shehory, O. (eds.) AAMAS. pp. 124–131 (2007)

[77] Schnoebelen, P.: The complexity of temporal logic model checking. In: Advances in Modal Logic. vol. 4, pp. 1–44 (2002)

[78] Searle, J.: The construction of social reality. Free Press, New York (1995)

[79] Singh, M.: A social semantics for agent communication languages. In: Dignum, F., Greaves, M. (eds.) Issues in Agent Communication. LNCS, vol. 1916, pp. 31–45. Springer (2000)

[80] Singh, M.: Formalizing communication protocols for multiagent systems. In: Veloso, M.M. (ed.) IJCAI. pp. 1519–1524 (2007)

[81] Singh, M.: Semantical considerations on dialectical and practical commitments. In: Fox, D., Gomes, C.P. (eds.) AAAI. pp. 176–181. AAAI Press (2008)

[82] Singh, M., Chopra, A., Desai, N.: Commitment-based service-oriented architecture. IEEE Computer 42(11), 72–79 (2009)

[83] Singh, M., Huhns, M.: Service-Oriented Computing: Semantics, Processes, Agents. Wiley, London (2005)

[84] Sirbu, M.: Credits and debits on the internet. IEEE Spectrum 34(2), 23–29 (1997)

[85] Spoletini, P., Verdicchio, M.: Commitment monitoring in a multi-agent system. In: Burkhard, H.D., Lindemann, G., Verbrugge, R., Varga, L.Z. (eds.) CEEMAS. LNCS, vol. 4696, pp. 83–92. Springer (2007)

[86] Spoletini, P., Verdicchio, M.: An automata-based monitoring technique for commitment-based multiagent systems. In: Hübner, J.F., Matson, E.T., Boissier, O., Dignum, V. (eds.) COIN. LNCS, vol. 5428, pp. 172–187. Springer (2009)

[87] Sultan, K., Bentahar, J., El-Menshawy, M.: Model checking probabilistic social commitments for intelligent agent communication. Applied Soft Computing 22, 397–409 (2014)

[88] Sultan, K., El-Menshawy, M., Bentahar, J.: Reasoning about social commitments in the presence of uncertainty. In: IEEE 12th International Conference on Intelligent Software Methodologies, Tools and Techniques. pp. 29–35 (2013)

[89] Telang, P., Singh, M.: Business modeling via commitments. In: Kowalczyk, R., Vo, Q., Maamar, Z., Huhns, M. (eds.) SOCASE. LNCS, vol. 5907, pp. 111–125 (2009)

[90] Telang, P., Singh, M.: Specifying and verifying cross-organizational business models: An agent-oriented approach. IEEE Trans. on Serv. Comp. 5(3), 305–318 (2012)

[91] Telang, P.R., Kalia, A.K., Singh, M.P.: Modeling health care processes using commitments: An empirical evaluation. PLoS ONE 10(11), 1–20 (2015)

[92] Telang, P.R., Singh, M.P.: Comma: A commitment-based business modeling methodology and its empirical evaluation. In: van der Hoek, W., Padgham, L., Conitzer, V., Winikoff, M. (eds.) Proceeding of the International Conference on AAMS. pp. 1073–1080 (2012)

[93] Torroni, P., Chesani, F., Mello, P., Montali, M.: Social commitments in time: Satisfied or compensated. In: Baldoni, M., Bentahar, J., van Riemsdijk, M.B., Lloyd, J. (eds.) DALT. LNCS, vol. 5948, pp. 228–243. Springer (2010)

[94] Venkatraman, M., Singh, M.: Verifying compliance with commitment protocols: Enabling open web-based multiagent systems. Autonomous Agents and Multi-Agent Systems 2(3), 217–236 (1999)

[95] Wan, F., Singh, M.: Enabling persistent web services via commitments. Information Technology and Management 6(1), 41–60 (2005)

[96] Weiss, G.: Multiagent systems: A modern approach to distributed artificial intelligence. The MIT Press (1999)

[97] Winikoff, M., Liu, W., Harland, J.: Enhancing commitment machines. In: Leite, J.A., Omicini, A., Torroni, P., Yolum, P. (eds.) DALT. LNCS, vol. 3476, pp. 198–220. Springer (2005)

[98] Wooldridge, M.: An Introduction to Multi-Agent Systems. John Wiley and Sons (2000)

[99] Wooldridge, M.: Semantic issues in the verification of agent communication languages. Autonomous Agents and Multi-Agent Systems 3(1), 9–31 (2000)

[100] Wooldridge, M.: An Introduction to Multi-Agent Systems. John Wiley and Sons (2009)

[101] Wooldridge, M., Jennings, N.: Intelligent agents: Theory and practice. Knowledge Engineering Review 2(10), 115–152 (1995)

[102] Xing, J., Singh, M.: Formalization of commitment-based agent interaction. In: SAC. pp. 115–120. ACM (2001)

[103] Xing, J., Singh, M.: Engineering commitment-based multiagent systems: A temporal logic approach. In: Proc. of the 2nd Int. Joint Conf. on AAMAS. pp. 891–898 (2003)

[104] Yolum, P., Singh, M.: Commitment machines. In: Meyer, J.J.C., Tambe, M. (eds.) ATAL. LNCS, vol. 2333, pp. 235–247. Springer (2002)

[105] Yolum, P., Singh, M.: Flexible protocol specification and execution: Applying event calculus planning using commitments. In: Proc. of the Int. Joint Conf. on AAMAS. pp. 527–534. ACM (2002)

[106] Yolum, P., Singh, M.: Reasoning about commitments in the event calculus: An approach for sepcifying and executing protocols. Annals of Mathematics and Artificial Intelligence 42(1–3), 227–253 (2004)

[107] Yoo, T., Jeong, B., Cho, H.: A petri nets based functional validation for services composition. Expert Systems with Applications 37(5), 3768–3776 (2010)