Flexible Agent Protocols via Temporal Linear Logic

A thesis submitted in fulfilment of the requirements for the degree of Doctor of Philosophy

Duc Quang Pham
B.Sc.

School of Computer Science and Information Technology
College of Science, Engineering, and Health
RMIT University

August 2007
Declaration

I certify that except where due acknowledgment has been made, the work is that of the author alone; the work has not been submitted previously, in whole or in part, to qualify for any other academic award; the content of the thesis is the result of work which has been carried out since the official commencement date of the approved research program; and, any editorial work, paid or unpaid, carried out by a third party is acknowledged.

Duc Quang Pham
School of Computer Science and Information Technology
RMIT University
31st August, 2007
Acknowledgments

While offering my substantial gratitude to Associate Professor James Harland and Associate Professor Michael Winikoff, my supervisors, I would like to express my appreciation of the supports from Professor Neil Furlong and Ms Kathryn Thomas at RMIT Research & Innovation Office as well as my family.
Credits

Portions of the material in this thesis have previously appeared in the following publications:


This work was supported by the Atlantic Philanthropy Foundation, RMIT Research and Innovation Portfolio, and RMIT School of Computer Science and Information Technology. The thesis was written in TeXnicCenter editor, and typeset using the \LaTeX\ document preparation system.

All trademarks are the property of their respective owners.
Contents

Abstract 1

1 Introduction 2
  1.1 Research Problem 2
  1.2 Background 3
  1.3 Our Approach 4
    1.3.1 Research Questions 4
    1.3.2 Approach 5
    1.3.3 Rationale 6
  1.4 Contributions 7
  1.5 Thesis Structure 8

2 Background 9
  2.1 Agents 9
    2.1.1 BDI Agents 11
  2.2 Flexible Protocols 12
    2.2.1 Agent Interaction Protocols 12
    2.2.2 Flexibility in Agent Interaction Protocols 16
  2.3 Commitments and Related Works 19
    2.3.1 Coordination 19
    2.3.2 Commitments 21
    2.3.3 A Model of Agent Negotiation on Commitments 25
  2.4 Temporal Logic - Computational Tree Logic* 27
    2.4.1 Temporal Logic 27
    2.4.2 Computational Tree Logic* (CTL*) 29
3.3.2 Base commitments ........................................... 71
3.3.3 Goals ......................................................... 72
3.3.4 Conditional Commitments ................................. 73
3.3.5 Pre-commitments ............................................. 75
3.3.6 Formation of Commitments by Reasoning and Negotiating Pre-commitments 76
3.3.7 Evolution of Commitments ................................ 78
3.3.8 Conventions .................................................... 79
3.3.9 Rules for Protocols ......................................... 79
3.3.10 State .......................................................... 80
3.3.11 Cricket Bat Sale Example ................................. 82
3.4 Summary ......................................................... 82

4 Specifying Flexible Interaction 83
4.1 Specification of Interaction ................................. 84
4.2 Designing Interaction Specification ....................... 85
4.2.1 The Specification Language ............................. 85
4.2.2 Designing Goals ............................................ 88
4.2.3 Designing Pre-commitments ............................. 88
4.2.4 Specification for Cricket Bat Sale Example .......... 90
4.3 From Specification to Execution ........................... 92
4.3.1 Interaction Model .......................................... 93
4.3.2 Message Format ............................................. 93
4.3.3 Request Messages ......................................... 95
4.3.4 Proposal Messages ........................................ 96
4.3.5 Fulfilling Conditions of Conditional Commitments 96
4.3.6 Fulfilling Goals and Base Commitments .............. 97
4.3.7 Utilizing Pre-commitments ............................... 98
Relevance of a Pre-commitment to a Goal or Base Commitment ... 98
4.3.8 Integration of Changes into Protocol Execution ...... 101
4.3.9 Cricket Bat Sale Interaction ............................ 102
4.4 Summary ......................................................... 112

5 Modeling Decisions about Choices 113
5.1 Desiderata for a Choice Calculus .......................... 114
CONTENTS

5.2 Modeling Decisions about Choices ........................................... 118
  5.2.1 Representative Choice of a Subformula ............................... 121
5.3 Choice Calculus ................................................................. 122
  5.3.1 Soundness and Completeness of the Choice Calculus ................. 129
5.4 Dealing with Choices and Changes .......................................... 137
5.5 Summary .................................................................................. 139

6 Partial and Concurrent Handling of Goals and Resources .......... 141
  6.1 Restrictions on the Logic Fragment ......................................... 142
  6.2 Dynamic Division of Formulas ................................................ 145
    6.2.1 Equivalence between a Formula and Its Split Ups .................. 151
  6.3 Dealing with Choices in Û ...................................................... 160
  6.4 Reasoning about Splitting Up ............................................... 162
  6.5 Distributed Concurrent Problem Solving .................................. 163
  6.6 Summary .............................................................................. 165

7 An Execution Model ................................................................. 166
  7.1 A Fragment of TLL for Execution ............................................ 166
  7.2 An Implementation Framework .............................................. 168
    7.2.1 Agent Reasoning .......................................................... 171
  7.3 Further Implementation Issues ............................................... 174
    7.3.1 Proof Search in TLL ......................................................... 174
    7.3.2 Mapping Different Agents' Time Systems ................................ 174
    7.3.3 Derivations among Conjunctions and Disjunctions in Resource Contexts 176
    7.3.4 Other issues .................................................................... 178
  7.4 Summary .............................................................................. 178

8 Examples and Discussion .......................................................... 179
  8.1 Concert Ticket Example .......................................................... 179
    8.1.1 Our Approach to Specifying Concert Ticket Interaction .......... 181
    8.1.2 Construction of Concert Ticket Interaction ............................. 183
    8.1.3 Sequences of Interactive Actions ....................................... 191
    8.1.4 Integration of Changes ..................................................... 200
    8.1.5 Summary of Interactions ................................................... 202
  8.2 Temporal Linear Logic (TLL) Modeling .................................... 203
B Implementation Remarks 258
B.1 Data Structure ...................................................... 258
B.2 Functions for Agent Interaction .................................... 260
  B.2.1 How to Fulfill (Partly) a Goal/Base Commitment .......... 260
  B.2.2 How To Make a Request ........................................ 261
  B.2.3 How To Make a Proposal ...................................... 263
  B.2.4 How to Respond to a Request .................................. 265
  B.2.5 How To Respond to a Proposal ................................. 266
  B.2.6 How to Resolve a Goal/Base Commitment .................... 268
B.3 Utility Functions .................................................. 270
  B.3.1 Functions on Rules .............................................. 272
  B.3.2 Functions for Commitments .................................... 273
  B.3.3 Functions for Choices ......................................... 273
  B.3.4 Functions for Formulas ........................................ 274
B.4 CD Player & Book Example .......................................... 287
  B.4.1 Specifying Protocol ............................................. 288
  B.4.2 Reasoning Prior to Interaction ................................. 289
  B.4.3 Construction of Interaction .................................... 290
  B.4.4 Sequences of Interaction ...................................... 293
  B.4.5 Discussion .................................................... 295

Bibliography 296
List of Figures

6.1 Structure of $\Gamma$ ......................................................... 146
6.2 Structure of $\Gamma \sim C$ ................................................... 147
6.3 Structure of $\hat{C}$ ......................................................... 148
List of Tables

3.1 Summary of Interaction Concepts Modeling ........................................ 77
4.1 Summary of Specification for Cricket Bat Sale Example ........................ 104
5.1 New Sequent Rules of the Choice Calculus ........................................... 140
6.1 Summary of Inter-dependencies on choices among split ups .................. 150
8.1 Summary of Specification Rules for Concert Ticket Example .................. 183
Abstract

Multi-agent systems have emerged as a new field that addresses issues among distributed, interconnected and intelligent systems such as conflicts of interests, global constraints, and sharing of resources. Our research addresses an important aspect of multi-agent systems study, specifying interaction protocols, that ensures proper coordination among agents’ activities.

Because traditional approaches to specifying protocols tend to pre-specify sequences of interactive behaviors that agents should exhibit, protocols are likely prone to failure when the operating environment, such as the Internet, changes quickly. Our work aims to specify protocols with a richer degree of flexibility in agent interaction and gives agents more autonomy over their interactive actions to cope with on-going changes.

In order to achieve that, we take a declarative approach to protocol specification and make use of the notion of commitment, which provides a mechanism for coordinating interactive behaviors among agents, as a structural element. We show how agents can reason about commitments and protocol actions, choices and changes to achieve the desired results. Such a reasoning system is based on temporal linear logic, which incorporates both temporal and resource-sensitive reasoning. We also provide logic mechanisms that enable agents to reason about partial handling of resources and commitments as well as various strategies on agents’ choices and changes from the environment. Our specification and execution frameworks are shown to include many features that increase flexibility in agent interaction.
Chapter 1

Introduction

Recently, software development has evolved toward the development of intelligent and interconnected systems working in a distributed manner. These systems are designed to handle complex tasks and work independently on behalf of humans, and are often implemented as agent systems. As they are used to represent human interests and objectives, agents usually have to deal with issues such as conflicts of interest, global constraints, and sharing of resources when taking actions or interacting with other agents. Accordingly, reaching agreement and coordinating actions with other agents are critical issues for agent systems. Multi-agent systems have been developed as a means of addressing these issues, in which the concept of computing is primarily a process of interaction [Woolridge, 2002]. This means that correctly designing interaction protocols is vital to the success of the applications.

1.1 Research Problem

In traditional approaches to specifying interaction protocols, such as those used in distributed computing or computer networks, protocols are typically specified by predefining the roles of the agents, which sometimes includes a very detailed specification of the interactive behavior of the agents. Agents have to follow pre-defined plans and interactions are usually subject to strict constraints, such as the order in which messages arrive. Often there is little real choice for agents to make in these protocols. The success of an interaction is often dependent on how well environmental changes can be predicted, and whether an appropriate pre-specified plan can be designed in advance.

However, in frequently changing environments like the Internet, changes are difficult to predict, and hence devising pre-specified plans is not feasible. Protocols specified via
traditional approaches are too rigid, and unlikely to lead to successful agent interactions. What is missing is the ability for the agent to reason about the appropriate interactive behaviour and adapt its sequence of actions accordingly. In other words, the problem we are addressing is to make interaction protocols more flexible so that agents have the ability to find the appropriate behaviour in a changing environment.

1.2 Background

As discussed by [Chopra and Singh, 2004], in order to make interaction more flexible, it is important that an agent is able to negotiate with other agents about what interactions should take place. Protocols can then be specified in terms of constraints on interactions, leaving agents with the autonomy to determine how to act subject to these constraints.

One way to specify the protocol constraints is to use a declarative approach based on logic.

Logic has been used as formalism to model and reason about agent systems and agent interaction. Various agent logic systems have been based on BDI logic [Rao and Georgeff, 1991], which models agents by using an intentional stance to describe agents’ behaviors using beliefs (B), desires (D) and intentions (I). However, there is a gap between these BDI logics and their implementations [Harland and Winikoff, 2004], which makes it difficult to use these logics in agent systems.

Among other logics, linear logic has a large body of well-developed proof theories which allow a direct computational interpretation of the logic [Alexiev, 1994]. Linear logic is also very well suited for modeling resources that are produced and consumed [Alexiev, 1994], updating processes [Girard, 1995] and concurrency [Girard, 1987b], which makes it a good candidate for modeling resource-conscious, concurrent interactions (as found in agent systems). Linear logic has been used in agent systems to support reasoning about cooperation in problem solving [Küngas, 2003], and to strengthen the links between proactive and reactive reasoning about actions [Harland and Winikoff, 2004]. The links between proof search and agent planning in linear logic have also been investigated [Jacopin, 1993]. In addition, agent negotiation has been modeled in terms of proof search using linear logic [Harland and Winikoff, 2002].

In modeling systems, time is a critical dimension to be addressed. However, while linear logic is a good candidate for modeling systems which are both resource-conscious and concurrent, it does not address time relationships directly. Therefore, a combination of temporal
Our Approach

1.3 Our Approach

1.3.1 Research Questions

In our work, we will address the following research questions:
1. What is an appropriate framework for specifying flexible interactions that naturally deal with resources with respect to time?

2. What is an appropriate execution framework for turning such specifications into flexible interaction?

3. What is a framework that enables agents to reason about their choices and changes from the environment to take advantages of opportunities and deal with exceptions?

1.3.2 Approach

We take a declarative approach to protocol specification, i.e. specifying what is to be achieved rather than giving a detailed specification of how it is to be done. Declarative specification of protocols provides agents with autonomy over their interactive behavior, in that once a particular commitment is specified, it is up to the agent to determine how it is to be fulfilled. Moreover, declarative specification of protocols allows an agent to utilise mean-end reasoning techniques over commitments, as the agent does for goals, states and actions. The notion of commitment, which is a key concept for coordination among agents [Jennings, 1993; Singh, 1997], can be used for both of these aspects.

In real world situations, achieving goals involves the consumption and production of resources over time. Given that linear logic [Girard, 1987a] can naturally model consumable resources [Girard, 1995] and temporal logic has operators suitable for describing time-varying behaviors [Emerson, 1990], we make use of a natural combination of them, temporal linear logic (TLL) [Hirai, 2000b].

We therefore propose using a fragment of TLL as the underlying logic system for specifying protocols. We investigate how various major interaction concepts like resource, resource exchange, resource transformation, capability, state, state update, goal, base commitment, and conditional commitment can be modeled in TLL.

Furthermore, proof search techniques in TLL can be used to provide reasoning capabilities for agents to help them determine what actions they should take. This reasoning could be testing whether a given property is true or not, or finding a series of actions that should be taken in order to achieve a given property. Another possibility is to commence with the current state, and apply inference rules to see what new states may occur. These reasoning methods correspond to proactive and reactive behaviors respectively [Harland and Winikoff, 2004]. Hence, proof search in TLL will be investigated to see how agents may reason about
commitments, the use of resources and interactive behaviors and handling environmental changes.

Specifications of protocols are defined as a set of potential conditional commitments (pre-commitments) for each agent and expressed in TLL. Our execution model then uses proof search in TLL as a mechanism for agent reasoning and a basis for converting specifications into interactive actions. In particular, agents interact by exchanging requests, proposals and responses to them, in which pre-commitments are offered as services to each other. Agents use the above-mentioned reasoning capabilities to find out how to utilize their resources, actions, capabilities, and pre-commitments to fulfill their goals, commitments and answer requests from others. They also use such reasoning to choose suitable pre-commitments about which to negotiate. Interactions are hence formed as commitments are resolved and new commitments emerge among agents.

We also extend the proof search methods of TLL in order to enable agents to reason about choices and environmental changes. In particular, we model the dependencies between choices and changes, and then develop appropriate inference rules for agents which allow them to apply a variety of strategies, such as as deciding choices in advance or delaying as long as possible, and taking either a bold or cautious approach to dealing with changes. In addition, we develop mechanisms in TLL to enable agents to partially handle resources, actions, goals and commitments as appropriate.

We then consider the strengths and weaknesses of the modeling and specification framework. Numerous examples are given throughout to facilitate discussion. In addition, given the existing links between TLL and timed Petri nets (TPNs) [Hirai, 1999], we further explore a theoretical base for a mapping between formulas of a fragment of TLL and states of TPNs. This mapping allows TPNs to be used as an execution model for specifications in TLL, and hence existing TPN techniques can be used for examining properties of execution. This approach, however, has some limitations that makes it less attractive than the approach that uses proof search.

1.3.3 Rationale

The agent paradigm is well-suited to dealing with complex systems consisting of many interacting components. However, traditional approaches to interaction protocol specifications are too rigid, which severely hinders the agents’ ability to interact. Recent approaches provide some of the required flexibility, but their underlying logic systems are quite limited when
it comes to the modeling of resources and time.

Our logic-based approach has many advantages. Firstly, logic is a well-known formalism which has often been used to model and reason about software systems. Secondly, the logic that we use is a combination of linear logic and temporal logic, and hence is advantageous to modeling resources, updates, concurrent and ongoing interactions. Thirdly, our approach aims to equip agents with reasoning abilities in the same logic that the protocol specifications are defined, which enables agents to reason directly about specifications. Lastly, interaction protocols are based on the concept of commitment, which also enables agents to reason about the protocols in a meaningful way. Overall, our approach provides a bridge between logic and multi-agent systems, using a logic well-suited to computation and utilizing the modeling of agent interaction protocols based on commitment. Accordingly, our framework will be able to support applications in electronic commerce, such as online auctions, automated travel agents, and the packaging of Web services.

1.4 Contributions

The thesis makes several contributions.

Firstly, modeling of various major interaction concepts using TLL has been achieved.

Secondly, a framework for declarative specification of protocols that centers on the notion of commitment is derived. The specification framework is shown to provide agents with autonomy and flexibility in interaction. Several methodological steps for designing protocol specifications are also established.

Thirdly, an inference system, choice calculus, is developed based on the TLL sequent calculus to provide agents with an ability to reason about (internal) choices and indeterminate possibilities and strategies in dealing with them. These strategies include deciding (internal) choices and indeterminate possibilities in advance or at their associated time, and taking a safe approach or a bold approach in dealing with indeterminate possibilities.

Fourthly, mechanisms for partial handling of goals, commitments, resources, and actions are also developed based on the results we derive concerning the splitting of formulas in a fragment of TLL.

Fifthly, an execution framework for such specifications in TLL that makes use of proof search in choice calculus is identified. This framework is also shown to bring more flexibility to agent interaction.

Lastly, the relationships between TLL and timed Petri nets are clarified by a formal result
that timed Petri nets are models of an intuitionistic fragment of TLL. This result makes it possible for converting specifications in TLL into timed Petri nets and hence existing TPN techniques can be deployed in the execution and verification of protocols.

1.5 Thesis Structure

The remainder of the thesis is structured as follows. Chapter 2 discusses background on agent interaction protocols, commitment-based approaches to agent interaction, linear logic, temporal linear logic, Petri nets and timed Petri Nets. Chapter 3 presents our modeling of agent interaction concepts. Chapter 4 presents a specification framework for interaction protocols and identifies an execution framework for such specifications. Chapter 5 discusses the modeling of choices and strategies on them. Chapter 6 applies the results developed in Chapter 5 to partial handling of goals, commitments, actions and resource formulas. Chapter 7 further examines a possible implementation of the execution framework introduced in Chapter 4. We then discuss the strengths and weaknesses of the modeling of interaction concepts and the specification and execution framework in Chapter 8. Chapter 9 presents our conclusions and possibilities for future work.
Chapter 2

Background

We begin our discussion of background work by discussing agents, BDI-model for agents, interaction protocols, commitments, temporal logic, linear logic, temporal linear logic and Petri nets.

2.1 Agents

Recently, software development has evolved toward the development of intelligent and interconnected systems working in a distributed manner. These systems can handle increasingly complex tasks and are often designed to work autonomously, i.e. without direct human oversight. Accordingly, often these systems have to deal with conflicts of interests or sharing of resources. They also need to be able to cooperate with other systems. Multi-agent systems have emerged as a new field that addresses these issues and reflects the concept of computing as primarily a process of interaction [Woolridge, 2002].

Intelligent autonomous agents are considered as a computer system situated in an environment, sensing its inputs and performing actions to modify the environment. Operating environments can be physical, i.e. in the real world, or be software environments, where agents’ actions change values of data and sensory inputs are the output of reading such data.

Intelligent autonomous agents embody these following properties [Wooldridge and Jennings, 1995a;b; Franklin and Graesser, 1997]:

- **autonomy**: agents exhibit their own control over their behaviors and internal states without human direct intervention.

- **reactivity**: agents respond to changes in their environment in a timely manner.
• **pro-activeness**: agents actively initiate actions to achieve their goals. This is a form of goal-directed behavior.

• **situated**: agents exert influences on the operating environment and perceive their effect.

• **social ability**: agents interact with other agents (and humans) to meet their objectives.

The importance of agent-based computing has been recognized widely in research and development of complex software systems [Jennings et al., 1998b]. [Jennings, 2000] argued that in modeling, designing and building complicated distributed software systems, the agent paradigm has significant potential. In fact, converging changes in system architecture, software engineering and human computer interface technology have shifted the emphasis from algorithms to interaction [Wegner, 1997]. Interactive systems are believed to be more powerful than algorithms in problem solving and agents are proposed as a new theoretical model of computation to reflect current computing reality more closely than Turing machines [Wegner, 1997].

Agents are often developed as parts of a multi-agent system. A multi-agent system contains many agents which may be of different designs, architectures, and objectives but interact with each other either as collaborators or competitors. The organisation of the individual agents into a multi-agent system can vary from being homogeneous to heterogeneous, from having a hierarchical control structure to a democratic one, and from independent to controlled execution of member agents [Huhns and Singh, 1998]. Multi-agent systems also differ in their approach to communication languages, protocols, specification of goals, beliefs, ontology and procedures.

Multi-agent systems are different from “traditional” distributed systems in two main aspects [Woolridge, 2002]. The first is that agents normally do not share common goals nor ultimately follow overall system objectives. Instead, they must follow such strategies that achieve their respective goals. Therefore, interactions among them are more like games. Secondly, autonomous agents are supposed to decide at run time how they coordinate their activities and cooperate with others in a dynamic manner rather than follow hardwired specifications.

The agent paradigm has become well-suited as a design metaphor to deal with complex systems comprising many components each having their own thread of control and purposes and involved in dynamic and complex interactions. Some examples include OASIS [Ljungberg
and Lucas, 1992] — an air traffic control system; ADEPT [Jennings et al., 1998a] — a business process management system, and the computer game Creatures [Grand and Cliff, 1998]. Applications of multi-agent technology have covered a wide range of important industrial areas such as information management, process control and electronic commerce [Moulin and Chaib-draa, 1996].

2.1.1 BDI Agents

There are many approaches to designing rational agents. A physical stance refers to modeling systems based on information about their physical states and natural laws. An intentional stance refers to modeling systems as an intentional system having beliefs, desires and intentions. A design stance takes into account information on functionalities of the systems. Among these, an intentional stance has been commonly adopted.

In an intentional stance, an agent’s behavior is based on its beliefs (B), desires (D) and intentions (I). In particular, beliefs represent information that an agent has about the world, which is often incomplete or may be in many cases incorrect. Desires indicate the states that agent wish to be brought about. Intentions are regarded as partial plans that the agent commits to carrying out to achieve its goals [Bratman, 1987]. Intentions are used as the causes for actions and involve consideration of future actions. Bratman argued that intentions are not reducible to beliefs and desires as agents have bounded resources and therefore rather than continually weighing up their competing desires and variable beliefs, they must settle on some particular course of action. Intentions act as constraints on the reasoning that agents use to select an action to perform. Also, the agents must have some form of commitment in order to coordinate and plan future actions [Cohen and Levesque, 1990]. In this way, intentions provide a screen of admissibility for other future intentions and act as a milestone for agents to measure the “success” of their attempts. In relation to desires, intentions are regarded as those desires that agents have chosen to commit to achieving [Wooldridge, 2000].

[Cohen and Levesque, 1990] proposed a formalism in which intention is defined as a temporal sequence of beliefs and desires while Rao and Georgeff developed a formalism for BDI systems which treats intention with equal importance as desires and beliefs and also specified the relationships among them [Rao and Georgeff, 1991].

Rao and Georgeff’s formalism utilizes a notion of possible-worlds for rational agents. While Cohen and Levesque address each possible world in a time-line structure [Rao and Georgeff, 1991], Rao and Georgeff represent them in a time-branching (tree like) structure in
Flexible Protocols

a similar manner to CTL* ([Emerson, 1990]). To account for the possibilities in the future, many possible-worlds are used to specify the possible circumstances of the environment. Options for agents to select are specified within each possible world. From each particular time point in a time tree-like structure that models the world, a set of worlds represents the worlds that agents believe to be possible. These are belief-accessible worlds. Similarly, there are desire-accessible worlds and intention-accessible worlds. Rao and Georgeff also specified an inter-relationship among these worlds and their compatibility, which helps to avoid unwanted side effects. They also formalized inner determinism (self) on the agents’ actions and outer determinism (decided by the world) on the outcomes of actions.

[Wooldridge, 2000] took a further step by adding a temporal component and an action component in the LORA framework. The resulting logic uses constructs built on top of classical logic and modal logic to capture properties and relations of agents, their beliefs, desires and intentions together with their actions and resulting effects in a temporal dynamic manner [Wooldridge, 2000].

However, as discussed in [Harland and Winikoff, 2004], there is a gap between BDI model theories and BDI implementations. Indeed, while BDI theories focus on relationships among possible worlds of beliefs, desires and intentions, most implemented BDI architectures are built around beliefs, events and plans.

2.2 Flexible Protocols

2.2.1 Agent Interaction Protocols

In single-agent systems, agents reason about choices for their own actions while considering changes in the environment. In multi-agent systems, agents are often self-interested, aiming to maximizing their own utility and achieving their own goals. Agents are also of different designs and come from different sources. One agent’s interests may be in conflict with others. Due to the presence of other agents, more possibilities are required in the agents’ reasoning. On the one hand, agents have to co-exist with each other as they share the same environment and many resources. On the other hand, they exercise their social abilities in such a way that they interact to take advantage of each other’s resources and capacities to achieve their respective goals more effectively or more efficiently than they could alone. Communication among agents enables them to coordinate their actions and behavior to achieve system overall coherence. Indeed, the ability to engage in high-level social interactions and to operate in flexible organizational structures is a fundamental requirement of agent systems [Jennings,
To understand how interactions can be constructed, we consider an approach [Kinny, 1999; d’Inverno et al., 1998] that deploys the notions of services and tasks. Services refer to the work provided by agents to meet requests of other agents or humans. When a request for services is sent, the other agent must either decline or accept. If it accepts, a service agreement is formed which imposes certain obligations on the involved parties. These obligations are decomposed into tasks. Tasks are smaller units of work which are part of the services and are assigned when agents accept a service request. Tasks are then the main medium for interaction during the provision of services. The involved agents may need to perform tasks for the execution of the service. During service execution, the requester of services may cancel the agreed services or suspend or resume them. When all tasks of the service have been carried out, the service agreement is fulfilled. If there are no pending requests or execution of services, the interaction ends.

Agent interactions are underpinned by communication protocols at various levels including the lowest level of interconnection, the level of format and syntax of exchange information, and the highest level, which is concerned with the meaning and semantics of the information. Once communication protocols are defined and agreed upon, interaction protocols can be used. Unlike communication protocols, interaction protocols are more concerned with a series of messages rather than the exchange of a single message. To be more specific, interaction protocols include structured sequences or patterns of interactive actions and involve two or more roles and exchanges of communicative messages among the agents. Usually, agents are presented with several alternative interactive actions of the protocol and they have to choose according to their own internal strategies. The protocols are, indeed, concerned with the perspective of the interaction rather than of a particular agent.

A growing area of research on Agent Communication Languages (ACLs) addresses the concerns regarding issues such as format of exchange messages, semantic models of these messages, conversation policies (or protocols), and shared ontologies and content languages [Kone et al., 2000]. In particular, issues regarding message format usually include primitive communicative acts and message parameters like sender, receiver, ID, and language. Communicative acts are defined in most traditional ACLs based on Speech Act theory [Searle, 1969], which reflects ideas from language philosophy. Moreover, making sure that messages are interpreted correctly and consistently relies on the semantics of ACLs. While many approaches to semantics are centered around agent mental attitudes (beliefs and intentions), others emerge from the concept of social agency [Singh, 1998]. Conversation policies which
Flexible Protocols

can be thought of as [Greaves et al., 1999] “general constraints on the sequences of semantically coherent messages leading to a goal”, are made up of pre-defined patterns of interaction that govern agent communication. Typical patterns include implementations of direct communication protocols, the contract net protocol, and the mediated communication protocol.

Several examples of ACLs include The Knowledge Query and Manipulation Language (KQML) [Patil et al., 1998], the ARTIMIS Communication Language (ARCOL) [Sadek et al., 1997], the FIPA standard ACL [FIPA, 1999], the InterAgent Communication Language (ICL) [Martin et al., 1999] and social semantics approaches like [Singh, 1998; Colombetti, 1998; 2000]. [Kone et al., 2000] provides a comprehensive review of ACLs. Among approaches to ACLs, increasingly popular are those based on the notion of social agency. As argued in [Singh, 1998], traditional approaches like KQML and ARCOL have several drawbacks. In particular, for example, KQML and ARCOL lack a public perspective in the meaning of their communicative acts, which hinders efforts to check for compliance with standardized meanings. Also, coverage of communicative acts (assertive, directive, commissive, permissive, prohibitive, declarative, and expressive) is very limited and hence it prevents agents from functioning effectively in complex social relationships. A commitment-based semantics approach to ACLs overcomes these drawbacks and promotes design autonomy and execution autonomy of agents [Singh, 1998].

Our approach also embraces the idea of social agency semantics for agent communication, although designing a particular ACL is beyond our scope of research. We instead focus on the higher levels of protocol interaction.

In a multi-agent environment, agents act under global constraints and must deal with issues such as conflicts over shared resources, deadlock, livelock, dependencies between agent actions, and a potential inability to accomplish system goals due to a lack of competence, resources or information. This makes it critical for agents to coordinate their actions with the anticipated actions of other agents. In particular, they must have the ability to to determine shared goals and common tasks, prevent unnecessary conflicts, and to accumulate and share knowledge [Huhns and Stephens, 1999].

The design of interaction protocols should observe the following criteria [Fornara and Colombetti, 2003]:

- interaction consists of legal sequences of communicative acts whose meanings are well-defined in an application-independent library of communicative acts in a standard agent communication language
accommodation of both purely reactive agents (which blindly follow the protocols) and deliberate agents (which reason about their actions)

• effective verification of the protocols' specifications is supported

In traditional approaches to specifying agent interaction protocols, such as those used in distributed computing or computer networks, protocols are often predetermined sequences of interactive behaviors with predefined roles for agents. Agents play their roles by following a pre-determined set of instructions. Any choices in these protocols are hardwired in advance with pre-specified conditions, leaving the agent no scope for any autonomous choice of its own. Accordingly, the success of these interaction protocols depends crucially on how well any changes in the environment can be predicted in advance.

Many formal description techniques have been used to specify these protocols. Finite State Machines are a well-known formalism which have been used in many context, and also to define agent interaction protocols [Barbuceanu and Fox, 1995; Kuwabara et al., 1995; Haddadi, 1995]. Formal specification languages like Z [Spivey, 1992] or those based on temporal logic [Finger et al., 1993; Fisher and Wooldridge, 1994] have also been used to formally specify protocols. For example, Z was deployed [d’Inverno et al., 1998] to specify protocols used for the Agentis agent interaction model [Kinny, 1999]. There are approaches that make use of Petri nets to specify protocols [Cost et al., 1999; 2000; Mazouzi et al., 2002]. Other approaches build bridges between agents and tools like standard UML to develop methods to specify agent interaction protocols [Lind, 2002] with a focus on activity diagrams.

An extension of UML was derived as Agent UML to model agent interactions [Odell et al., 2001]. Agent UML uses three layers which contain templates and packages to represent the overall protocol and various principal types of diagrams to capture both intra-agent and inter-agent dynamics. Another approach applies modifications to many aspects of UML such as protocol diagrams, agent roles, extended UML message semantics and nested and interleaved protocols [Bauer et al., 2000]. Others recast description techniques for Agent UML interaction diagrams using UML 2.0 [Huget and Odell, 2004] or extend UML with features appropriate for electronic commerce and supply chain management [Huget, 2002].

Given a specification of an interaction protocol, it is important to verify that agent interactions actually conform to the specification. Several techniques have been designed to address the verification task with respect to specifications. Verification can be done at design time, at run time or after the interactions finish. Approaches to ensuring that agents respect interaction protocols vary in the way the specification is given and the verification
Flexible Protocols

Techniques used. In one approach, formal specification techniques for interaction protocols are used as a basis for verification of protocols [d’Inverno et al., 1998]. In another approach, model checking is utilized to verify specifications [Wen and Mizoguchi, 1999]. Model checking is based on explicit enumeration of all possible computation paths to check for satisfaction of desirable system properties, and then returns either that all properties are satisfied or a counterexample. Some approaches are based on theorem proving techniques [Fisher and Wooldridge, 1997], which makes use of the underlying logic of the system. Others [Endriss et al., 2003] represent protocols as if-then rules which can be checked for conformance in advance or enforced at run time, based on the logical interpretation of the system, and not the dialogue history and conditions related to the agents’ private knowledge base.

However, traditional approaches to protocol specification specify protocols which are too rigid for agent systems. Predefined sequences of interactive behaviors demand that only actions that are within the specifications are legal. As a result, agents cannot adopt new actions or alter their interactive action sequences to deal with changes. Hence, when critical changes in the environment take place frequently, these protocols can quickly become obsolete.

2.2.2 Flexibility in Agent Interaction Protocols

To cope with frequent and unexpected changes in operating environments, protocols should be designed not only to capture the legitimate sequences of actions but also to be altered at run time as appropriate. Specification of protocols should accommodate various run time options and allow agents to reason about appropriate actions and cope with ongoing changes. In other words, protocols should be flexible. In more concrete terms, the desirable flexibility of protocols is characterized [Yolum and Singh, 2002b] as:

- ensuring preservation of an agent’s autonomy in its interactive behaviors, and constraining agents only to the extent necessary to run the protocol;
- accommodating agents of different architectures, designs and strategies; and
- allowing agents to explore arising opportunities to make a better choice or simplify the interaction and handle exceptions that result from unexpected behaviors of other agents or from the environment.

The first item addresses the vital property of agents as being autonomous in the context of structured interaction. Rigid protocols are those that dictate interactive behaviors for agents
Flexible Protocols

what and when to carry out a pre-determined course of action. Flexible specifications permit agents to follow their respective strategies and internal deliberation as far as possible and do not constrain agents beyond the core nature of the interaction they describe.

The second item is concerned with how specification of protocols is not designed for a particular design group of agents but generally enough for agents having numerous different designs and/or architectures. Also, protocols should be specified without imposing constraints on the set of strategies that agents can deploy. Our approach however does not focus on achieving this second criterion, although we do not make any particular assumptions about our agents’ design, architecture, or set of interaction strategies.

In order to achieve flexibility in interaction protocols, several approaches have been used. For example, one approach provides formal specifications of agent interaction which are a part of a wider concept of electronic institution [Esteva et al., 2001]. The approach considers agents’ roles and the meanings of exchange messages but to a limited extent.

Another approach [Cheong and Winikoff, 2006a] aims to achieve flexibility and robustness in interaction by using the Hermes methodology for goal-oriented design of interaction. Instead of pre-defining sequences of interactive actions, the protocols designers can specify high level concepts like goals, available actions and temporal constraints. Interactive messages then emerge as agents decide which actions to take to fulfill the interaction goals while satisfying the constraints. The Hermes methodology starts with identifying the roles of agents and interaction goals. Interaction goals are refined into a hierarchy of sub-goals with temporal constraints (dependencies) among them. Roles in interaction are then assigned to these sub-goals to determine the participation of agents. Actions that are available to achieve goals and their temporal constraints are determined for each goal, which is reflected in action maps. While the action maps show all possible execution sequences, action sequence diagrams are used to capture possible sequences of actions and can be used to check for coverage of action maps with respect to the goal hierarchy. Interactive messages can be determined between actions in action sequence diagrams. Action failures can be handled by retrying the actions or selecting another possible action. Goal failures can be handled by terminating the current interaction and/or rolling back to a previous goal. This handling of goal failure can be specified in terms of permissions for termination and/or rollback of each goal. Furthermore, design artifacts are mapped into collections of plans for agents [Cheong and Winikoff, 2006b]. Such collections contain coordination plans for coordinating the agents’ actions via rules on temporal constraints and relationships among subgoals in the goal hierarchy; achievement plans for pursuing interaction goals; and interface plans for converting interaction messages
into events and goal events for the agents’ internal processing. Overall, the Hermes methodology adds a certain degree of flexibility and robustness to agent interaction by allowing interaction messages to emerge as agents act to fulfill interaction goals under constraints and by the inclusion of failure handling mechanisms [Cheong and Winikoff, 2006a].

[Kakas et al., 2004; 2005] provides a framework which represents protocols at two levels — normal and exceptional sets of preferred interactive actions in accordance with the contexts of interaction. Various context-dependent protocol versions captured in a uniform theory can cover different aspects of the protocol under different circumstances or as perceived by different agents, which offers a high degree of flexibility. The authors went further to model a theory that describes individual strategies of agents. Strategies of agents are modeled considering a variety of situations in which interactions can take place as well as the dynamically changing circumstances of the interactions. This modeling helps agents to exhibit adaptable behaviors according to the interaction contexts and their particular roles. The two theories, one describing public protocols and the other describing individual agent strategies, can then be unified under one expressive representation framework. The overall decision of which actions to use is based on a consideration of both theories via argumentation-based reasoning that depends on the contexts, which, as a result, provides flexibility in agent interaction.

The interplay between research in formal dialectics and in multi-agent systems has demonstrated its potential via much recent work in the field of dialectic models of protocols [Reed, 1998]. Research in dialectics is concerned with study of dialectical contexts within which arguments are suggested [Hamblin, 1970]. In particular, different kinds of fallacies involved in argumentation as well as formal notions and tools that deal with them are studied [Maudet and Chaib-draa, 2002]. Two different approaches that share most of the dialectical principles of interaction are dialog-game based approaches and commitment-based approaches.

Approaches that are based on the notion of commitment define and attach meanings to the agents’ interactive actions, communicative acts or ACL using social commitments ([Jennings, 1993; Singh, 1998; 2000]) to make protocols more flexible. In particular, Yolum and Singh utilized the notion of social commitments to capture the meanings of states and actions of protocols, which, as a result, allows agents to reason about them during interaction. They view interactions as exchange and manipulation of commitments in a commitment machine [Yolum and Singh, 2002a]. Another approach that also defines interactive actions as operations to create or manipulate commitments among agents makes use of event calculus to formalize the domain-independent reasoning rules and operations on commitments [Yolum and Singh, 2002b; 2004]. These approaches based on social commitments make protocol
execution more dependent on the state of the conversation than on the previous messages [Singh, 2000]. They provide agents with the ability to follow their commitments, but the agents are not obliged to, which hence adds a certain level of flexibility. We will discuss these approaches further in Section 2.3.2.

Nevertheless, many commitment-based approaches to flexible protocols have certain limitations. Firstly, commitment machines require prior specification of the commitments to be processed during interaction [Yolum and Singh, 2002a]. Agents, however, should be allowed to negotiate the commitments by themselves. Secondly, commitment machines do not provide a mechanism for agents to acquire resources or capacities by exchanges with other agents. Thirdly, the underlying logic is centered on temporal and classical logic, which are not natural for modeling resources.

While the commitment-based approach utilizes the notion of social commitment to define the semantics of agent communication languages or interactive actions to model protocols with flexibility, the dialog-game based approach can be regarded as an extension of it, with a treatment of protocols as agent communication structures [Maudet and Chaib-draa, 2002]. Several such approaches have been developed [Reed, 1998; Dastani et al., 2001; McBurney and Parsons, 2002; Maudet and Evrard, 1998; Chaib-draa et al., 2003], and a critical review of these works can be found in [Maudet and Chaib-draa, 2002].

Overall, a declarative logic based approach to protocol specification, i.e. what is to be achieved rather than how to achieve it proves several advantages. A declarative approach allows explicit declaration of rules composing protocols. It also supports formal verifications of some properties of the protocols [Singh, 2000] like safety, and liveness. Some examples of declarative approaches to specifications include using a fragment of linear logic — LO [Andreoli and Pareschi, 1990], Forum [Miller, 1996] and [Kanovich et al., 1998].

Our approach is to specify protocols in a declarative manner. The specification also uses the concept of commitment to capture the meaning of interaction goals, states and actions to allow mean-ends reasoning over them. The next section describes the concept of commitment and related literature in detail.

2.3 Commitments and Related Works

2.3.1 Coordination

In distributed and decentralized environments containing many self-interested agents, often there are dependencies among the agents' actions, conflicts between shared resources, and
global constraints. To work effectively, agents need to coordinate their activities with each other [Jennings, 1996]. Coordination is a process in which agents reason about their own actions and anticipated actions of others and try to enforce coherence to the behaviors of the community. Approaches to provide coordination mechanisms include ([Jennings, 1996]):

- organizational structuring, which defines the pattern of information and control relationships (authority) among participating entities [Cockburn and Jennings, 1996; Shoham and Tennenholtz, 1992; Decker, 1996];

- exchanging meta-level information, which refers to sending control level information about current priorities and focus to each other, such as Partial Global Planning [Durfee, 1996]; and

- multi-agent planning, which details plans for participating agents about their future actions and interactions to achieve their respective objectives, which can be either centralized [Georgeff, 1983; Cammarata et al., 1988] or decentralized [Corkill, 1979; Rosenschein and Genesereth, 1988].

Another approach is to use human teamwork models. In particular, like human beings, agents need to trust that others are performing their allocated roles and use a governing system (conventions) which manage unacceptable behaviors [Fornara and Colombetti, 2002]. When humans work together in a team, their mental states have vital roles in coordination [Levesque et al., 1990; Cohen and Levesque, 1991]. The use of intentions creates stability and predictability in the agents’ actions [Woolridge, 2002]. Moreover, having individual intentions toward a common goal and having a kind of collective intention toward that goal differ in that the latter exerts a certain level of responsibility toward the other members of the team [Levesque et al., 1990]. Hence, intentions embody aspects that are of vital importance to agent interactions.

Breaking down the mechanism of intention, [Cohen and Levesque, 1990] derived the preliminary importance of interpersonal commitments. Prior to the work, [Fikes, 1982] indicated that in an informal domain like office work, its functioning relies heavily on the formation, negotiation, satisfaction, monitor and discharge of workers’ commitments to one another. [Winograd and Flores, 1987] also recognized social commitments as foundation for social interaction in such environments. Later, [Jennings, 1996] also discussed how commitment underlines intention. It went further to argue that commitment and its governing mechanism (convention) lay the foundation for coordination among agents in multi-agent systems.
2.3.2 Commitments

“All coordination mechanisms can ultimately be reduced to commitments and their associated conventions” - Jennings [Jennings, 1993].

We investigate the use of the notion of commitment to facilitate agent interaction.

Commitments refer to strong promises of the agents to undertake some course of action. Commitments hence are mainly concerned with future actions. For example, a sales agent is supposed to deliver the goods once the customer has paid. The agent makes a promise that it will deliver the goods in time whenever customers pay. This promise is expected to be kept under standard conditions and hence will affect the agent in its subsequent behaviors. This promise is captured by the concept of a commitment of the sales agent, which is advertised to customers as an incentive to make payments.

Agents can have internal commitments involving only themselves [Bratman, 1987; Cohen and Levesque, 1990], social commitments made from one agent to another agent to carry out a certain course of action [Castelfranchi, 1995; Singh, 1999] or joint commitments to an overall goal [Levesque et al., 1990]. Singh further analyzed the roles of internal commitments (psychological commitments) and social commitments [Singh, 1996]. Internal commitments arise within agents as in their intention to achieve certain conditions and do not require the agents to be liable to others. Social commitments emerge as agents interact and hold agents accountable to others and hence contribute to coordination and coherence of agents’ activities. Social commitments are a powerful abstraction mechanism for agent interaction.

The role of commitment has been recognized as significant in agent interaction. Indeed, persistence of commitments offers a certain level of stability [Wooldridge and Jennings, 1994] and predictability, which is especially useful when agents deal with issues of inter-dependencies, global constraints or resources sharing [Jennings, 1993]. Commitments allow other agents to make assumptions about the agent’s actions and determine their own actions accordingly. Hence, the notion of commitment helps agents to overcome the uncertainty caused by the distribution of control among agents, which is typical in multi-agent systems [Jennings, 1993].

There have been approaches treating commitments as psychological constructs [Grosz and Sidner, 1990; Levesque et al., 1990] which are related to BDI models of agents and social constructs [Singh, 1997]. The former captures persistence in agent intentions and relies on
mutual beliefs among agents. For example, cancellation of commitments is approached by demanding that participating agents must reach a mutual belief that the commitments have been cancelled [Singh, 1997]. However, mutual beliefs require agents to hold beliefs about each others’ beliefs to an unbounded level of nesting, which is quite impractical without simplifying assumptions, especially when communication is not guaranteed [Singh, 1996]. The latter approach, using social constructs, argues that social constructs are not dependent on psychological constructs [Singh, 1997]. Social commitments bind the committing agents to other agents or a community and makes a distinction between debtor and creditor of the commitments.

While the notion of commitment imposes constraints on the agents’ subsequent behaviors, conventions provide a monitoring mechanism for commitments, such as under which circumstances commitments are reassessed, and then retained or corrected or broken [Jennings, 1993]. In fact, conventions determine the degree of flexibility and reactivity of agents’ behaviors [Wooldridge and Jennings, 1994].

Moreover, in conventions, the constraints on commitments are specified in accordance with characteristics of the operating environment. In particular, how often an agent re-assesses its commitments depends on the characteristics of the environment such as the frequency of changes, and the degree of uncertainty. The corresponding penalties and remedies for an agent’s breaches or rectification of commitments depend on the degree of trust, interdependencies and the level of complexity present in the environments of interactions [Jennings, 1993]. Conventions indeed provide the flexibility necessary for cooperating agents to cope with changes in dynamic environments [Jennings, 1996].

Much work has been focused on using the concept of commitment to model agent interaction.

Commitment is well recognized via the interplay between commitments and the structure of multi-agent systems [Singh, 1999]. This also shows that many other concepts like obligations, taboos, and conventions can be captured as different kinds of commitments.

To implement agent interaction, Jennings argued the use of commitments and conventions as a foundation [Jennings, 1993]. Commitment has been recognized as an essential mechanism for coordinating activities among cooperative agents [Sen and Durfee, 1994]. Indeed, explicit interactions among agents base their meaning on commitments among communicating agents [Singh, 1999]. Commitment has been used to express the meaning of communication messages among interactive agents, whether it is defined operationally using the object-oriented paradigm [Fornara and Colombetti, 2002] or as a construct in a logic system [Yolum and
Singh, 2002b]. Interaction protocols can be defined in this way, based on the meanings of communicative acts [Fornara and Colombetti, 2003; Flores and Kremer, 2004].

Resolving conflicts among commitments within agents then becomes essential to ensure coordination among the agents’ activities. Effective handling of commitments among agents requires timely dissemination of information on commitments, keeping participating agents aware of relevant conventions as well as knowledge about interdependencies among commitments [Jennings, 1993].

Haddadi developed a theory that takes an internal perspective of agents to reason about their behaviors, based on BDI architectures which focused on the interplay between beliefs, desires and intentions [Haddadi, 1995]. The theory extends to reasoning about cooperating with other agents using commitment as the fundamental construct for interaction. An agent to agent commitment is one that once it is reached between a pair of agents, it becomes a mutual promise to be followed by both agents. Commitments between two agents are formalized in terms of the conditions under which commitments are formed and maintained. Various stages of commitment formation such as determining potential for cooperation, making pre-commitments and making commitments are also formalized. A framework in which agents can reason about these stages to form interaction is provided [Haddadi, 1995].

Several other approaches define the semantics for agent communication languages and communicative acts (and to a variety of extents, interaction protocols) based on social commitments. In the ALBATROSS framework, [Colombetti, 2000] defined the semantics of agent language based on social commitments in a temporal logic based on CTL* ([Emerson, 1990]). It also recognized the notion of pre-commitment, which, when accepted by the agent it is addressed to, becomes a commitment. Another approach [Fornara and Colombetti, 2003] makes use of commitment-based meanings of exchange messages and allows the specification of the preconditions and effects of the performance of communicative acts and verification of protocol’s soundness according to criteria that are related to the meaning of the exchanged messages. Similarly, the approach [Flores and Kremer, 2002] further emphasized that commitments are also the binding factors of the conversations via the notion of conversational commitments.

The notion of social commitment has also been used to attach meanings to agents’ interactive actions and states in a declarative manner and allowed protocols to be captured via the concept of a commitment machine [Yolum and Singh, 2002a]. A commitment machine is defined as the possible states of the interaction, operations for state transitions and possible final states. These states and actions have declarative semantic content in terms of commit-
ments. In particular, a state is specified by those commitments that are in force. Actions are determined by the effects they have on commitments. New states can then be logically inferred from the meanings of current states and actions performed.

Hence, rather than specifying protocols as sequences of interactive actions, their approach specifies protocols in terms of laying out the meanings that are legal in the protocol and the final meanings. Executing a protocol then becomes reaching a desirable state by logical inference of actions that alter the meaning of intermediate states appropriately. The use of commitments make the protocol more flexible as it allows diverse sequences of interactive actions as long as they can reach desired final states. The commitment machine approach also has the advantage that commitment machines can be formally compiled into deterministic finite state machines which are sound and complete with respect to the computations allowed by the commitment machines.

This line of work on commitment machines is furthered by an attempt to provide the ability to reason about cancellation of commitments under exceptional cases. For example, a commitment to pay is canceled when the goods delivered are damaged and returned to the company. The attempt uses non-monotonic causal logic [Giunchiglia et al., 2004] to allow such defeasible reasoning about commitments [Chopra and Singh, 2004] instead of classical logic in which it is quite cumbersome to implement such an ability.

Another approach to specifying interaction protocols as operations on commitments uses event calculus [Kowalski and Sergot, 1986] to formalize the reasoning rules and operations on commitments [Yolum and Singh, 2002b]. Event calculus has been shown to have advantages for representing traditional network protocols clearly and precisely, when compared with traditional approaches using process algebra [Denecker et al., 1996]. Moreover, to reach a higher degree of flexibility of agent interactions, rather than specifying conditions or fixed penalties for breaking commitments via conventions, agents are provided with a reasoning ability such as weighting the ongoing cost of executing current commitments at various stages together with penalties for replacing commitments [Excelente-Toledo et al., 2001].

Other works address several issues concerning the use of the notion of commitment to facilitate agent interaction [Amgoud et al., 2000b;a; Guerin and Pitt, 2001]. One approach [Mallya et al., 2003] represents explicitly the time constraints of commitments using Computational Tree Logic [Emerson, 1990] on commitment constructs and provides reasoning about those temporal aspects in a domain-independent manner. Another [Xing and Singh, 2003] also addresses the handling of time constraints of commitments and further identifies several commitment patterns and agent behavior models and their compatibility. Normal interactive
Conversations of agents are converted into commitments and their relationships are identified in order to construct behavioral models of the participating agents which can successfully capture commitment-level interaction protocols [Wan and Singh, 2003]. Yet another [Sen and Durfee, 1994] also argues for the use of commitments to deal flexibly with dynamic environments. Adaptive strategies for making commitments in the domain of distributed task scheduling are also investigated in relation to environment factors.

For applications in the domain of e-commerce, a meta-model for inter-operation that is particularly suitable for flexible agent interactions in e-commerce was proposed [Xing et al., 2001]. Moreover, commitment-based specification is also applied to business processes in Web applications based on protocols [Singh et al., 2004]. A methodology and software tool for specifying interaction as rule-based commitment protocols using the Web Ontology Language (OWL) and the Semantic Web Rule Language (SWRL) were also derived to facilitate business process designers [Mallya et al., 2005].

In summary, flexibility of protocols can be obtained by the use of the notion of commitment in several aspects. Commitments bring meaning into agents’ goals and interactive actions. Moreover, commitments facilitate coordination as they motivate agents to perform expected behaviors. Our work reinforces the use of commitments to model interactions flexibly.

2.3.3 A Model of Agent Negotiation on Commitments

The process of means-end reasoning is reasoning about how to achieve a goal state. It can be expressed as reasoning about “chance + choice + commitment” [Rao and Georgeff, 1993]. From the perspective of BDI agents, reasoning about choices means reasoning about the decisions that an agent can make based on what it believes possible and what it desires to achieve. Chances refer to the possibilities brought by the environment. An agent’s commitment to itself or to another agent expresses situations in which the agent should take on a chosen course of action and under what conditions it should release the commitment.

[Haddadi, 1995] considered commitments between agents in the form of joint intentions [Cohen and Levesque, 1991; Rao et al., 1992] to achieve joint goals. Taking an internal perspective, the work defines a joint goal as a goal that is delegated to another agent or adopted from another agent. Commitments in this setting are referred to as mutual commitments in which both agents agree on the terms of cooperation and in the case of BDI agents, also mutually believe that they both want to commit.
Engagement of two agents in a mutual commitment can be achieved in two phases: a pre-commitment phase and a commitment phase [Haddadi, 1995].

In the pre-commitment phase, agents exchange proposals indicating what they are willing to commit. The proposals are in a form of pre-commitment that shows an agent is willing to commit to doing an action or achieving a state (under some conditions) if the other agent also wants it to. A pre-commitment is not an commitment but a willingness to commit and requires an agreement from the other party before it can become a commitment.

Haddadi’s work further considers two cooperative contexts — task delegation and task adoption [Haddadi, 1995]. Task delegation means delegating achievement of a goal to another agent, while task adoption means adopting achievement of another agent’s goal. Task delegation and adoption are essential aspects of the cooperative problem solving process among agents.

Pre-commitments can then be classified into pre-commitments for task delegation and pre-commitments for task adoption. We denote a subscript “d” for task delegation and “a” for task adoption. Given ϕ is a TTL formula, pre-commitments are [Haddadi, 1995]:

- \( \text{Pre-commit}_d \alpha \beta \varphi \) which means that agent \( \alpha \) pre-commits to delegating to agent \( \beta \) the task of achieving \( \varphi \).
- \( \text{Pre-commit}_a \beta \alpha \varphi \) which means that agent \( \beta \) pre-commits to adopting the task of achieving \( \varphi \) for agent \( \alpha \).

With similar denotation as above, mutual commitments can be classified as ([Haddadi, 1995]):

- \( \text{Commit}_d \alpha \beta \varphi \) means that agent \( \alpha \) commits to choosing agent \( \beta \) to achieve \( \varphi \).
- \( \text{Commit}_a \beta \alpha \varphi \) means that agent \( \beta \) commits to achieving \( \varphi \) for agent \( \alpha \).

To respond to a proposal of a pre-commitment, the other agent can agree or disagree to the proposal or propose another pre-commitment. Negotiation can be carried out in this manner on pre-commitments.

In the two contexts of task delegation and adoption, to accept a proposal of a pre-commitment of task delegation, an agent needs to make the corresponding pre-commitment of task adoption and vice versa. The other agent can then indicate its acceptance by informing its respective pre-commitment or its later derived commitment. For example, when agent \( \alpha \) makes a pre-commitment of delegating the task \( \varphi \) to agent \( \beta \) (\( \text{Pre-commit}_d \alpha \beta \varphi \)), it
also requires the agent $\beta$ to make a pre-commitment of adopting the task $\varphi$ from agent $\alpha$ 

$\text{Pre}_{\text{commit},a} \beta \, \alpha \, \varphi$.

A pre-commitment can turn into a commitment when itself and its corresponding pre-commitment of task delegation/adoption are both evident to both parties [Haddadi, 1998]. When two pre-commitments turn into commitments, the process moves to the commitment phase. In the commitment phase, both parties are engaged in a mutual commitment on the task via their respective commitments on the task’s delegation/adoption. The two agents then actually commence performing the relevant actions.

Having discussed the role of commitment in agent interaction and commitment-based approaches to achieve flexibility in interaction, we now turn to work describing the use of logic as a formalism for modeling interaction.

### 2.4 Temporal Logic - Computational Tree Logic*

#### 2.4.1 Temporal Logic

Temporal logic (TL) can be regarded as a formal system which addresses the description and reasoning about the changes in truth values of logic expressions over time [Emerson, 1990]. Temporal logic is a special class of modal logic ([Lewis, 1918]). One of the main themes of use of TL is for reasoning about concurrent programs. Temporal logic introduces temporal operators like “sometimes”, “always”, “next” and “until” to encode the time relationships among logic formulas. In particular, “$\Diamond \ A$” means $A$ is true sometimes, “$\square \ A$” means that $A$ is always true, “$\Diamond A$” means that $A$ is true from the next time point, and $A \cup B$ asserts that $A$ is true until a time when $B$ is true.

There are different kinds of temporal logic systems. A review of their different kinds and their applications can be found in [Emerson, 1990]. Temporal logic systems can be classified along various dimensions — as either propositional or first order, as global or compositional, as branching or linear, as points-based or intervals-based, as discrete or continuous and as referring to past tense or future tense. Propositional TL is the natural extension of classical propositional logic while first-order TL contains expressions constructed from variables, constants, functions, predicates, and quantifiers. There are “global” TL systems that allow global reasoning about a complete concurrent program and also “compositional” TL systems that allow reasoning about program fragments and combining correctness proofs of constituent sub-programs (as lemmas) into a proof of the whole program. The treatment of time in TL systems can be linear where at each moment, there is only one future moment,
or branching (tree-like nature), where there may be several different possible alternative futures for each moment. Moreover, when temporal operators are evaluated regarding truth values against points in time, the TL systems are referred to as points-based systems. In contrast, intervals-based TL systems have temporal operators evaluated over periods of time, which may simplify checking certain correctness properties. The structure of time as discrete – where consecutive moments correspond to consecutive program states – or continuous – such as containing real numbers rather than non-negative integers – is also a differentiating factor. Lastly, temporal operators of TL systems can be intentionally regarded as describing past or future occurrences of events, corresponding to past tense or future tense TL systems respectively. Most TL systems for reasoning about concurrency are future tense where they consider only the starting state and its (future) subsequents. However, the inclusion of past tense operators can make specifications more natural and expressive [Lichtenstein et al., 1985]. Overall, research has been concentrated on global, points-based, discrete time and future tense TL systems [Emerson, 1990].

Regarding its contribution to reasoning about multi-agent systems, temporal logic was proposed in 1977 [Pnueli, 1977] as a tool for reasoning about reactive systems. For example, it is quite elegant to express in temporal logic a typical property of reactive systems such as “if a request is sent, then a response is eventually given”. Temporal logic operators are quite suitable for describing and reasoning about continuously operating programs or non-terminating computations for ongoing interactions which are referred to as reactive systems [Emerson, 1990].

Moreover, temporal logic can be also used for the specification and verification of concurrent and reactive programs [Emerson, 1990]. In these systems, the principle task is maintaining an ongoing interaction with the environment in which intermediate outputs of the systems might be used as feedback to subsequent intermediate inputs to the systems. As the systems can continuously operate for an arbitrary long period or be non-terminating, final states generally cannot be determined. Hence, approaches that use initial state/final state semantics to specify the system behaviors are no longer adequate. Temporal operators like sometimes or always, however, can be suitable for asserting reactive systems’ behaviors and properties.

In verifying of such concurrent and reactive systems, correctness properties like liveness and safety are typically of major concern [Pnueli, 1977; Emerson, 1990]. Safety or invariance properties demand that each finite prefix of a (possibly infinite) computation satisfies certain requirements. Liveness properties require that some finite prefixes of a computation
meet requirements a certain number of times. Informally, safety properties states that “nothing bad happens” while liveness properties express the notion that “something desired will (eventually) happen”.

Verification can then be carried out in two main ways – proof-theoretic and model-theoretic [Emerson, 1990]. The heart of the proof-theoretic approach is making use of a formal deductive system of TL which contains axioms and inference rules while manually composing the program’s correctness proof for a certain TL specification. The advantage of this approach lies in human intuition that guides the proof search; the disadvantage is that carrying out the proof manually is often difficult and tedious. On the other hand, the model-theoretic approach automates the tasks of program construction and verification by using decision procedures to manipulate the corresponding temporal models of the program and specifications. A decision procedure takes a TL specification formula P and decides if P is satisfiable or not. In particular, for the case of protocols, which are often considered as finite state concurrent systems, model checking is more likely to be done automatically in practice. In fact, when the global state transition graph of finite state concurrent systems is treated as a finite temporal logic structure, a model checking algorithm can be applied to check if the structure is a model of a TL formula representing a certain correctness specification and hence, meets the requirement of the specification. Compared with the proof-theoretic approach, the model checking approach seems to be more widely applicable in practice. For example, model checking was successfully used for a mutual exclusion problem [Clarke et al., 1986], and in the areas of designing sequential circuits [Browne et al., 1985; Dill and Clarke, 1990].

2.4.2 Computational Tree Logic* (CTL*)

CTL* refers to an infinite tree-like branching structure of time [Emerson, 1990]. CTL* is a more expressive extension of CTL [Clarke and Emerson, 1982]. In CTL*, the past is regarded as deterministic with a linear structure and the future has a branching structure. CTL* takes into account the uncertainty of the future by associating possibilities with branches. Formulas have their values either determined at each time point or remained static along some future paths. As CTL* has a branching structure for time rather than a linear structure, it has greater expressiveness, such as allowing quantifying operators over branches to be followed by an arbitrary time formula of linear structure [Emerson, 1990]. It also allows Boolean combinations and nesting over temporal operators, which makes CTL* in effect a “full”
branching time logic [Emerson, 1990].

While temporal logics are suitable for modeling temporal properties and relationships, when it comes to modeling systems with a resource-conscious view, linear logic and its extensions seem to be an appropriate choice. In the next section, we will discuss linear logic and its use in modeling systems.

2.5 Linear Logic

2.5.1 Linear Logic and Its Connectives

In real life, resources are consumed to bring about new resources or to achieve objectives and once being consumed, the resources are no longer available. In logics such as classical or temporal logic, however, a direct mapping between resources and formulas is likely to cause trouble. In fact, formulas are interpreted as truth values and throughout their existence, they can be used again and again, infinitely, in derivations of other formulas. In other words, by their truth value interpretation, they inherently demonstrate a sense of unlimited resources.

In particular, when we consider an implication \( A \Rightarrow B \), if \( A \) is true, then we can derive \( B \). After the derivation, \( A \) remains unchanged and hence can be reused. If we make a direct association of classical logic formulas with resources, for example, we denote:

- \( A \) as “having one dollar”,
- \( B \) as “having a chocolate bar”,

then it can be read for \( A \Rightarrow B \) as “having one dollar leads to having a chocolate bar”.

The problem is that the implication allows us to get a chocolate bar while we still have the one dollar. The one dollar can then be reused to buy another chocolate bar. This process can be repeated countlessly to get unlimited chocolate bar while retaining the one dollar. Though this problem can certainly be overcome by careful modeling using extensions of classical logic, such a direct and simple mapping between formulas and resources as above is difficult to obtain.

In providing such simple and natural mapping between resources and formulas in linear logic, Girard proposed the constraints according to which every formula should be used exactly once and its copies should no longer be freely added (duplication) or removed (discard) in logic derivations [Alexiev, 1994]. In the above example, such an implication in linear logic like \( A \to B \) will allow \( A \) to be removed in deriving \( B \), which means that after using one dollar
to buy a chocolate bar, the dollar is gone, hence, providing a natural treatment of formulas as resources. Indeed, as a linear implication can not be iterated due to that its conditions are consumed (or modified) after use, it can truly reflect the sense of causal relationship in real life.

As a result of these constraints, linear logic [Girard, 1987a] (LL) was introduced by Girard in 1987 as a resource-conscious logic.

Deduction in LL is typically based on sequent calculus rules. In particular, a logical consequence can be written in a form of a sequent: \( A \vdash B \) where \( A \) is the antecedent and \( B \) is the succedent. Sequent calculus rules provide ways of transforming sequents while preserving deductibility. A proof or a derivation of a goal sequent can be then constructed via a sequence of applications of sequent calculus rules which transform a set of system axioms into the conclusion sequent. If a proof of a sequent can be regarded as a way to verify a logical consequence based on a set of axioms, then LL sequent calculus rules are regarded as strategies for the verification.

In terms of sequent calculus, at the heart of the differences between linear logic and classical logic (CL) and most other extensions of CL is the abandonment of two structural rules of CL — contraction and weakening.

**Contraction Rule - classical logic**

\[
\frac{\Gamma, \varphi, \varphi \vdash \Delta}{\Gamma, \varphi \vdash \Delta} \quad \frac{\Gamma \vdash \varphi, \varphi, \Delta}{\Gamma \vdash \varphi, \Delta}
\]

**Weakening Rule - classical logic**

\[
\frac{\Gamma \vdash \Delta}{\Gamma, \varphi \vdash \Delta} \quad \frac{\Gamma \vdash \Delta}{\Gamma \vdash \varphi, \Delta}
\]

In a common sense reading, the contraction rule suggests that if we can buy one of the goods using several \( \varphi \), then we can buy it with just one \( \varphi \). The weakening rule says that if we can buy one of the goods with \( \Gamma \), then it is acceptable to spend some more \( \varphi \)'s without getting anything more in exchange. Hence, weakening allows causes without effect. In a more formal sense, from a bottom up reading, these rules correspond to the mechanisms of duplication and discarding of formulas respectively.

As a result of removing two classical structural rules, while sequents in CL comprise of sets of formulas which are expandable or contracted via duplication and merging, sequents in LL are composed of multi-sets of formulas in which the number of occurrences of formulas is
important. Moreover, the idempotent property of logic connectives is also adjusted [Alexiev, 1994] — only some connectives (additive connectives) remain idempotent.

Another result is that classical conjunction (and) and disjunction (or) were recast subject to different uses of contexts (passive formulas) — multiplicative use as combining and additive as sharing. Multiplicative means that contexts are totally separated and must be concatenated in the conclusion of inference rules that involves a use of these contexts. Additive means a complete sharing of contexts and requires these contexts to be the same. [Girard, 1995] described the resulting four logic connectives:

1. **Multiplicative conjunction** $\otimes$ (times) with neutral element 1, for example $A \otimes B$, means that one has both A and B at the same time.

2. **Additive conjunction** $\&$ (with) with neutral element $\top$, for example $A \& B$, stands for one’s own choice either of A or B but not both.

3. **Additive disjunction** $\oplus$ (plus) with neutral element 0, for example $A \oplus B$, stands for the possibility of either A or B, but we don’t know which.

4. **Multiplicative disjunction** $\boxtimes$ (par) with neutral element $\bot$, for example $A \boxtimes B$, means that if not A then B or vice versa but not both A and B.

There is a distinction between two conjunctions $\otimes$ and $\&$. Though both conjunctions express the availability of two objects, but with $\otimes$, we have both, and with $\&$, we have only one, according to our choice. For example, given one dollar we can buy a chocolate bar: $\text{dollar} \rightarrow \text{choc}$ or we can buy a cup of tea: $\text{dollar} \rightarrow \text{tea}$. $\text{dollar} \rightarrow \text{choc} \otimes \text{tea}$ means that we can buy both with only one dollar, which is not right. On the other hand, $\text{dollar} \rightarrow \text{choc} \& \text{tea}$ means that using one dollar, we can buy only one, a chocolate bar or a cup of tea and we have a choice.

Multiplicative disjunction $\boxtimes$ is also different from additive disjunction $\oplus$. $A \boxtimes B$, which can be defined as $A^\bot \rightarrow B$ or $B^\bot \rightarrow A$, requires that if A is present then there must be a negation of B (as a cause for the presence of A) or vice versa. On the other hand, $A \oplus B$ requires that one is present while a negation of the other may or may not be present. Hence, as compared to $\oplus$, $\boxtimes$ further exhibits a dependency between two objects A and B.

Regarding $\&$ and $\oplus$, apart from the apparent difference between a conjunction and a disjunction in an additive context, there is also a distinction between them with respect to the choice made. With $\&$, the choice is internally determined while with $\oplus$, the choice is externally determined.
Negation in LL $(\cdot)\perp$ still enjoys the property of being involutive as in classical logic, i.e. $A\perp\perp = A$, while being constructive. The involutive property makes the De Morgan laws applicable to all connectives and quantifiers. For example, $(\alpha \otimes \beta)\perp$ is equivalent to $\alpha\perp \& \beta\perp$. Moreover, linear negation was referred to by Girard as expressing duality or a change of standpoint. An action of type $A$ is a reaction of type $A\perp$. With a use of negation, linear implication $A \rightarrow B$ is defined as $A\perp \& B$.

In addition, in order to regain the power of the two structural rules — contraction and weakening — they are re-introduced in a different form. In fact, contraction and weakening are maintained in a more controlled manner by having them applicable only to formulas marked by the exponential “!” (of course) and its dual “?” (why not).

**Contraction Rule - Linear logic**

$$\frac{\Gamma, \varphi, !\varphi \vdash \Delta}{\Gamma, !\varphi \vdash \Delta} \quad \frac{\Gamma \vdash \varphi, ?\varphi, \Delta}{\Gamma \vdash ?\varphi, \Delta}$$

**Weakening Rule - Linear logic**

$$\frac{\Gamma \vdash \Delta}{\Gamma, !\varphi \vdash \Delta} \quad \frac{\Gamma \vdash \Delta}{\Gamma \vdash ?\varphi, \Delta}$$

The connective “!” suggests the iterability of an action or the absence of any reaction (corresponding to that action). For example, $!A$ means to spend as many $A$s as one may need. ! recovers the notion of infinite use of an object in classical logic. For example, implication in classical logic $A \Rightarrow B$ is modeled as $!A \rightarrow B$ in linear logic.

Regarding its expressiveness, linear logic is more expressive than classical logic. Indeed, the exponentials of linear logic allow it to represent the classical meaning of logic formulas. The linear connectives further add more expressive power to linear logic as they can express features like updating, and concurrency which demand more complicated handling in classical logic.

The expressiveness of linear logic, however, comes at the price of its computational complexity in deciding provability of linear logic formulas. A survey of the computational complexity of various fragments of linear logic can be found in [Lincoln, 1995]. Some important results are that the full propositional linear logic is undecidable and the propositional multiplicative-additive fragment of linear logic (MALL) which excludes exponentials ! and ? is PSPACE-complete [Lincoln et al., 1992]. The propositional multiplicative fragment of linear logic which contains only the connectives $\otimes$ and $\&$, their respective constants and
propositions is NP-complete [Kanovich, 1991]. So is the Horn fragment of LL [Kanovich, 1992]. Even a very limited part, constant-only multiplicative fragment of linear logic, still remains NP-complete [Lincoln and Winkler, 1994; Kanovich, 1994].

However, the high complexity of LL does not hinder much of its wide applicability [Alexiev, 1994]. For example, regarding logic programming, various restrictions can be implemented on the set of allowable formulas and proof search space, such as the work by Andreoli on focusing proofs [Andreoli, 1992]. Also, undecidability is not a barrier in the areas of applications where proof search is not of concern. This is the case when a fragment of linear logic is applied as a functional programming language whose logical content of functional programs correspond to proofs and computation corresponds to proof reduction, not proof search. Examples of applications of linear logic or its fragments in the area of functional programming languages include the work [Abramsky, 1993] which gives a computational interpretation of linear logic based on the Curry-Howard isomorphism (or “formulas as types” paradigm) [Howard, 1980]; the Linear Abstract Machine [Lafont, 1988], and LILAC [Mackie, 1994], a functional programming language.

2.5.2 Proof Theory

Linear logic has the desirable Cut Elimination property [Alexiev, 1994].

\[
\frac{\Gamma \vdash F, \Delta \quad \Gamma', F \vdash \Delta'}{\Gamma, \Gamma' \vdash \Delta, \Delta'} \quad \text{cut}
\]

In the proof search process, the cut rule makes computation difficult as its premises contain a formula which is not present at the conclusion, which consequently forces the search procedure to make a guess. The Cut Elimination property allows proofs that use the Cut rule to be turned into those without using it or into those that have all the instances of it pushed up to the leaves of the proof trees. In the latter case, leaves of these proof trees are logical axioms or non-logical axioms which can be handled by existing indexing techniques. Therefore, the property makes the proof search more efficient.

Moreover, the sub-formula property can be made applicable in linear logic. The sub-formula property states that if there is a proof of a formula then there is also a proof of that formula in which all formula occurrences are sub-formulas of the conclusion and assumptions. This property can significantly reduce the set of formulas to be tried in the search. Moreover, proof can be, to some extent, encoded directly into the formula to be proved [Andreoli and Pareschi, 1991], which corresponds to “syntax-directed proof search”.
An investigation [Andreoli, 1992] into proof-theoretical properties of linear logic further reveals an effective way for proof search [Alexiev, 1994]. Connectives of linear logic can be classified based on whether they demand a choice to be made in the proof procedure. A wrong choice may lead to dead-ends and backtracking. The connectives $\&, \otimes, ?,$, and the quantifier $\forall$ require no choice to be made and hence decomposition of the goal formulas can be carried out immediately. The connectives $\otimes, \oplus, !$, and the quantifier $\exists$ require a choice. The choices involved with these connectives can be of which way to split the context, of which of the two disjuncts to try on, of whether to remove the exponential, and of which concrete term to assign to the variable respectively. If the goal contains only formulas of these connectives then one of these formulas is selected for the proof to focus on. This technique of focusing proof pays less attention to proof permutations on the order of inference rules applied. Also, focusing proofs are complete with respect to full linear logic, which means if there is a general proof, then it has a corresponding focusing proof. A language LinLog is also provided as a natural base for focusing proofs, together with its mapping from full linear logic. Moreover, a constraint-based approach to deal with partial information in proof construction was introduced in relation to a formulation of “focusing” property [Andreoli, 1992].

Well-developed proof theories of LL allow a direct computational interpretation of the logic [Alexiev, 1994] and constructive proofs of satisfiability, and hence our approach to interactions can closely link the specification of protocols with the execution of them.

### 2.5.3 Advantages of Linear Logic with respect to Modeling Systems

Comparing to classical logic and other extensions of it, linear logic has the advantage of taking the number of occurrences of formulas into account. This, together with the ability to model production and consumption of resources, makes it a natural basis for applications in the areas such as object-oriented programming, databases, and natural language parsing [Alexiev, 1994]. As also remarked by [Girard, 1995], compared to (classical) logic programming, linear logic programming is advantageous with respect to updates, inheritance and parallelism.

A particular area in which linear logic is more advantageous than classical logic is modeling concurrency and parallelism [Girard, 1987b]. The multiplicative connectives of linear logic can be used directly as a means of expressing parallelism [Girard, 1987a]. The theoretical basis for concurrency based on linear logic has resulted in the development of Linear Objects [Andreoli and Pareschi, 1990]. Linear Objects (LO) enjoys properties of both logic
and object-oriented programming, such as declarative reading of programs, dynamic changing of object states, and structured representation of knowledge via built-in inheritance. Built-in inheritance, which means that methods of the “super class” objects are still applicable in the specialized objects, stems from the way methods can be invoked. Applicability of an object method is determined by having all of the logic literals of its head matched with a subset of the literals of the current object states. Consequently, adding more attributes (literals) to the object states (specialization) does not make these “superclass” methods inapplicable. Moreover, not only do LO clauses make use of $\otimes$ but also they are enriched with $\&$ in their body. While $\otimes$ can express external concurrency, the use of $\&$ can simulate parallel running of independent threads, which is referred to as internal concurrency [Bozzano et al., 2004]. Furthermore, Linear Objects was extended [Miller, 1996] to incorporate abstraction as well.

Linear logic manifests its suitability to modeling concurrency also via its tight correspondence with various models of concurrency, such as Petri nets [Brown, 1989; Martí-Oliet and Meseguer, 1989; Engberg and Winskel, 1993; 1994], Process Calculi [Kobayashi and Yonezawa, 1993a;b], and Chemical Abstract Machines [Andreoli et al., 1993].

Process Calculi are concerned with interactions among independent processes and provide a tool for manipulation and analysis of process descriptions as well as formal reasoning about them. ACL (Kobayashi and Yonezawa, 1993a;b) is a framework that models concurrent computation based on asynchronous communication. In ACL, computation is treated as proof construction in linear logic and mechanisms for concurrent computation like message passing, asynchronous communication, and identifier creation are also expressed in terms of linear logic.

The Chemical Abstract Machine (CHAM) [Berry and Boudol, 1990] treats a system of distributed processes as a chemical solution which contains floating molecules and membranes dividing the solution into a hierarchy of sub-solutions. When two matching molecules get in contact, a chemical reaction occurs following reaction rules and consequently consumes some of their parts and creates products. A major point of the CHAM concept is that interactions among disjoint sets of molecules can be in parallel. The Interaction Abstract Machine (IAM) [Andreoli et al., 1993] extends this notion further, and is also based on linear logic. IAM works on interactions among independent, and locally defined sub-systems, which moves “from pure chemistry to an elementary form of social-biology”. The concurrency is indeed dealt at two levels in IAMs — the level of competitive agents and the level of agents’ components.

At the core of many of these systems is the notion of update which is inherent to linear
logic. In classical logic, a theory about a system is normally modeled by the use of classical logic and axioms. They, however, remain unchanged throughout, which hinders its ability to model updates. Linear logic, in contrast, allows an explicit separation of current state as linear information, which is available for only a single use [Girard, 1995]. Hence, after use, information on the current state will be erased, giving way to new information. Transitions from old states to new states are realized by means of linear implication. For example, if $S \rightarrow S'$ is provable from the transitions, taken as axioms, then state $S'$ is accessible from state $S$. Given state $S$ and the transition $S \rightarrow S'$, the system updates from state $S$ to state $S'$. As a result, in linear logic, the updating or revision process can be performed naturally by means of logical consequences between states. The four connectives $\otimes$, $\&$, $\oplus$ and $\oplus$ also make linear logic more natural for modeling real life relationships. $\otimes$ allows a natural expression of proportion. For example, if A encodes “having one dollar” then by $A$ and $A$, we mean having two dollars. In classical logic, $A \land A = A$, which steals away one dollar. Linear logic overcomes this problem by using $\otimes$ for “and”, hence $A \otimes A \neq A$ and two $A$ are available for use.

[Harland and Winikoff, 2002] also emphasized that the difference between internal and external choices is natural in linear logic. For instance, we can specify that the choice of places A or B for the delivery of goods is made by the supplier (us) as “Place A” $\&$ “Place B” and made by the customer as “Place A” $\oplus$ “Place B”.

A good example of the suitability of linear logic to naturally model choices in computations is an analog between the computer instruction IF...THEN...ELSE and the use of additive conjunction $\&$ [Girard, 1995]. In particular, consider two transformation actions $A \rightarrow B$ and $A \rightarrow C$, and an instruction IF $\alpha$ THEN $A \rightarrow B$ ELSE $A \rightarrow C$. The computations that follow this instruction require making the choice between the THEN part and ELSE part based on the condition $\alpha$ which is determined internally in the computations up to the point of consideration. Likewise, in order to perform an action $A \rightarrow B$ $\&$ $C$, it requires a choice between the two actions $A \rightarrow B$ and $A \rightarrow C$ to be decided internally so that the selected one is carried out.

Furthermore, the duality exhibited by linear negation can be natural for modeling dual sides such as input/output, answer/question, consumption/production, etc. For example, in modeling agent interaction, positive formulas can be regarded as what is already provided while their negations can be treated as what is required. Moreover, as two sides of a linear implication can be considered as dual to each other, like consumption $\rightarrow$ production, formulas can be moved from one side to another by being converted to their linear negated forms. This
allows reasoning about state transitions to take place with the owing resources/formulas be carried to be the next state.

### 2.5.4 Applications of Linear Logic to Agent and Multi-agent Research

In multi-agent environments, resources are distributed among agents and resources produced by some agents can be supplied to other agents and in turn, will be consumed for further production. Hence, the ability to match consumption and supply of goods among agents can help them to cooperate and negotiate. [Harland and Winikoff, 2002] has pointed out that linear logic is a natural way to represent this mechanism. Indeed, negotiations among agents on who supply what can be supported by searching among linear logic representations.

[Küngas, 2003] also emphasized the use of a set of linear formulas to model a set of consumable resources which represents states of agents and to use linear implication to model transitions among states. Moreover, capacities of agents, which drive the transformation of resources, can also be modeled via linear implication. A deduction mechanism, Partial Deduction [Komorowski, 1981], was also introduced and adjusted particularly to the setting of linear logic [Küngas and Matskin, 2005]. Partial deduction, regarded as an optimization technique in logic programming, partially evaluates logic programs to derive a more specialized program which is more efficient for execution while keeping the meaning of the original program unchanged. Soundness and completeness of partial deduction techniques have been defined and proved [Küngas and Matskin, 2005; Küngas, 2004a] in the !HLL fragment around the notion of “executability” of applications of computation specification clauses which are essentially based on an extra-logical axiom. !HLL is the Horn fragment of linear logic that contains the \( \otimes \) and \(!\) connectives. Partial deduction can be applied to help agents to determine the missing capabilities they require to reach their goals and the appropriate trade with other agents to supply the missing capabilities. Negotiation among agents based on these trades can then be performed as a form of distributed cooperative problem solving. However, the authors made an underlying assumption that all agents are altruistic, i.e. that they will supply maximum resources whilst requiring minimal resources in return, which is quite restrictive, as agents are often self-interested.

The work [Harland and Winikoff, 2004] suggested modeling agents in a logic programming framework based on linear logic which covers both planned actions and reactive behaviors of agents. While planned actions of agents can be realised by standard backward-chaining techniques, reactive behaviors are likely to demand forward-chaining techniques which can
be based on standard sequent calculus of linear logic [Harland et al., 2000]. Informally, backward-chaining techniques start with a goal, identify intermediate required premises and work backwards until axioms are found. Backward-chaining techniques correspond to goal-directed proofs in which inference rules are chosen in favor of reducing the goal formulas rather than reducing the program clauses. On the contrary, a forward-chaining technique begins with axioms and applicable rules and results in new formulas. The combination of backward-chaining and forward-chaining techniques results in a mixed mode computation in linear logic, which provides a firm basis for the reactive and proactive characteristics of autonomous agents.

As discussed above, Interaction Abstract Machines [Andreoli et al., 1993] can be used to conceptually model concurrent interactions among multiple agents, via broadcast communication, and within agents, as blackboard systems. Both forms of communication can be implemented via message type asynchronous communication. The main restriction is the assumption that all agents speak the same language.

A specification language $\mu$ACL [Pérez, 2002] for agents was derived based on linear logic, extending the Asynchronous Communications in linear logic (ACL) language [Kobayashi and Yonezawa, 1993a:b], which is intended for concurrent and distributed logic programming. It is claimed that $\mu$ACL can naturally express essential aspects of agent programming such as communication, concurrency, resources, state update, modularity and inferences [Pérez, 2002]. Likewise, another linear logic programming language Lygon [Winikoff and Harland, 1995; Winikoff, 1997] was investigated for adaptation to agent programming [Amin, 1999].

Another approach is a theoretical foundation for algorithmic verification techniques for specifications based on a fragment of linear logic, which is Linear Objects (LO) [Andreoli and Pareschi, 1990] extended with universally quantified goal formulas [Bozzano et al., 2004]. The underlying mechanism is bottom-up evaluation of specifications based on effective “fix point operator” and a finite symbolic representation of a potentially infinite collections of first-order provable goals. Potential applications of this approach are in proving properties or verification of specifications of protocols and concurrent systems such as multi-agent systems. Examples have been developed that prove properties of specifications of mutual exclusion protocols in multi-agent systems [Bozzano, 2002] and automatically verify authentication protocols in cryptography [Bozzano and Delzanno, 2002].
2.5.5 Interpretations of Linear Logic

Formulas in linear logic can be interpreted as either resources or processes. We will discuss this duality as a starting point for our work on encoding agent interactions in temporal linear logic.

While linear logic propositions can be used to encode a resource item or a process of consumption or production of resource, its connectives can deal with relationships among resources or among processes. Examples of these interpretations are below. Note that we interpret multiplicative disjunction \(\AA\backslash\bb\) as \(\AA\bot\rightarrow\bb\) or \(\bb\bot\rightarrow\AA\).

**Formulas as Resources**

The interpretation is based on the work of Lincoln [Lincoln, 1992].

- **Multiplicative conjunction \(\otimes\) (times):** \(\AA\otimes\bb\) means that both resources \(\AA\) and \(\bb\) are available concurrently.
- **Linear implication \(\rightarrow\) (times):** \(\AA\rightarrow\bb\) means that a use of the resource \(\AA\) causes the resource \(\bb\) to be available.
- **Additive conjunction \& (with):** \(\AA\&\bb\) stands for one’s own choice of either resource \(\AA\) or \(\bb\) to be available but not both.
- **Additive disjunction \(\oplus\) (plus):** \(\AA\oplus\bb\) stands for the possibility of either resource \(\AA\) or \(\bb\) being available, but we don’t know which.
- **Linear negation (¬):** \(\AA\bot\) means a negative resource (debt) that cancels \(\AA\).
- **Modality (!):** \(\AA\!) means that we have as much of the resource \(\AA\) as we want.
- **Modality (?):** \(\AA\?) means that we demand as much of the resource \(\AA\) as possible (potentially an infinite amount).

**Formulas as Processes**

The interpretation is based on work [Mitchell, 1995]. A combined formula via a connective is treated as a process that produces resources.

- **\(\AA\otimes\bb\)** means a process that can produce exactly \(\AA\) and \(\bb\) and then halts.
• A → B means a process that takes as input A (consumes A) and produce as output B then halts.

• A & B stands for a process which has two buttons labeling A or B. If we press button A, the process will produce exactly A before halting, similarly for pressing B.

• A ⊕ B stands for a process covered by a black box contains either a process to produce A or a process to produce B.

• A⊥ means a process that consumes exactly A then halts.

• !A means a process that can produce A(s) as many times as required to satisfy other processes that consume A.

• ?A means a process that can consume A(s) as many times as provided by other processes that produce A.

Sequent Calculus

Treating formulas as resources, we present some informal interpretations of some relevant sequent calculus rules, based partly on [Alexiev, 1994] and [Girard, 1995].

Negation Rule

\[ \Gamma, A \vdash \Delta \]
\[ \Gamma \vdash A^\perp, \Delta \]

As linear negation (.)⊥ means a transposition or a change of standpoint, the rule means a change from a view of the process of producing A to a view as of consuming A.

\[ \Gamma \vdash A, \Delta \]
\[ \Gamma, A^\perp \vdash \Delta \]

This means a change from a view of a process as consuming A to a view as producing A.

Right multiplicative conjunction rule

\[ \Gamma \vdash A, \Delta \]
\[ \Gamma' \vdash B, \Delta' \]
\[ \Gamma, \Gamma' \vdash A \otimes B, \Delta, \Delta' \]

We can produce both resources A and B concurrently by combining the separate productions of each resource.

Right additive disjunction rule

\[ \Gamma \vdash A, \Delta \]
\[ \Gamma \vdash B, \Delta \]
\[ \Gamma \vdash A \oplus B, \Delta \] or \[ \Gamma \vdash A \oplus B, \Delta \]
Using either a process of producing $A$ or producing $B$, we can come up with a process of producing $A \oplus B$. This rule can be thought as a process of hiding – putting $A$ or $B$ into a black box and afterward only knowing that it must contain one of $A$ or $B$, but we cannot tell which one from the box.

**Left additive disjunction rule**

$$\Gamma, A \vdash \Delta \quad \Gamma, B \vdash \Delta$$

$$\Gamma, A \oplus B \vdash \Delta$$

When using $A \oplus B$ to produce $\Delta$, which $A$ or $B$ is available is not our choice. Therefore, we must have previously the capabilities to use either of them to derive $\Delta$.

**Right additive conjunction rule**

$$\Gamma \vdash A, \Delta \quad \Gamma \vdash B, \Delta$$

$$\Gamma \vdash A \& B, \Delta$$

In order to produce the outcome $A \& B$, which allows us to choose either $A$ or $B$ to be available (but not both), we must have concurrently the capability to produce $A$ and the capability to produce $B$.

**Left additive conjunction rule**

$$\Gamma, A \vdash \Delta \quad \text{or} \quad \Gamma, B \vdash \Delta$$

$$\Gamma, A \& B \vdash \Delta$$

When consuming $A \& B$ to produce $\Delta$, the choice of whether $A$ or $B$ to be used is ours, therefore, we need only one capability that uses $A$ (or $B$).

**Left linear implication rule**

$$\Gamma \vdash A, \Delta \quad \Gamma', B \vdash \Delta'$$

$$\Gamma, \Gamma', A \rightarrow B \vdash \Delta, \Delta'$$

We can chain a process that produces $A$ together with a process that consumes $B$ in separate contexts into one with concatenated context in which $A$ can be replaced by $B$.

**Left modality ! rule**

$$\Gamma, A \vdash \Delta$$

$$\Gamma, !A \vdash \Delta$$

When a single $A$ is used to produce $\Delta$, with a capability to supply $A$ as many as required, we can still produce $\Delta$. 
Temporal Linear Logic

Right modality ! rule

\[ \Gamma \vdash A, ?\Delta \]
\[ \Gamma \vdash !A, ?\Delta \]

If there is an unlimited supply of resources to produce A, then we can also produce as many A as desired.

Weakening for modality !

\[ \Gamma \vdash \Delta \]
\[ \Gamma, !A \vdash \Delta \]

Given an extra capability to supply A as many as required (or no supply at all), we can still produce \( \Delta \).

Left modality ? rule

\[ !\Gamma, A \vdash ?\Delta \]
\[ !\Gamma, ?A \vdash ?\Delta \]

In a context of unlimited supply (and/or demand) or resources, if a single A is consumed in the process then as many A as are available can also be consumed.

Right modality ? rule

\[ \Gamma \vdash A, \Delta \]
\[ \Gamma \vdash ?A, \Delta \]

When a process satisfies a demand for a single A, that process also satisfies a demand for as many as A as are available.

Weakening for modality ?

\[ \Gamma \vdash \Delta \]
\[ \Gamma \vdash ?A, \Delta \]

Given an extra demand of as many A as are available, we can still produce \( \Delta \).

2.6 Temporal Linear Logic

Linear logic has been extended to deal with time relationships [Kanovich et al., 1998; Kanovich and Ito, 1997; Tanabe, 1997; Hirai, 2000b].

Time is introduced [Kanovich et al., 1998] as a unary predicate like \( Time(t) \), which means the time is \( t \). Events and actions are associated with their corresponding unary time predicates by means of linear connective \( \otimes \), which implies concurrent co-existence among them. The author claimed that the approach deals with time qualitatively and quantitatively and also provides an easy and transparent way of specifying real-time finite state systems. However, rather than relying purely on inference rules such as the sequent calculus, its form of reasoning about time requires different treatments for time predicates and timeless predicates as well as external handling of time variable (such as arithmetic operations).
Linear logic has also been extended [Kanovich and Ito, 1997] with the temporal operators “next” \( \bigcirc \), “sometime” \( \Diamond \), and “later” \( \odot \). “Later” is a restricted form of “sometime” which does not include the present. Logical formulas are then regarded as specifications of sets of processes connected with time points. The author claimed that the resulting hybrid system is comprehensive with respect to describing temporal behaviors of resource-sensitive concurrent processes.

Another approach [Tanabe, 1997] also integrates temporal logic and linear logic. Formulas in the hybrid system are of two types, temporal formulas and (marking) formulas. Formulas of these two types can be combined together in the form \([X]A \text{ or } < X > A\), where \(X\) is a temporal formula and \(A\) is a (marking) formula. The former type behaves like intuitionistic propositional linear logic formulas with exponentials except that they have an inverse operator. Formulas of the latter can be preceded by modal operators to express temporal aspects. They are governed not only by standard intuitionistic linear sequent calculus but also by specialized axioms from modal logics and cut rules on temporal formulas. The resulting system can be used as a specification tool for certain classes of timed Petri nets.

However, as pointed out in [Hirai, 2000a], the former approach [Kanovich and Ito, 1997] does not include the modal operator \(!\), hence is not an extension of linear logic. Also, the latter approach [Tanabe, 1997] includes non-logical axioms and does not have a completeness theorem for timed Petri nets.

In contrast, temporal linear logic (TLL) [Hirai, 2000b] naturally extends linear logic with an inclusion of temporal logic S4 and the resulting logic system has the cut elimination property. Given the strengths of temporal logic and linear logic, temporal linear logic can be more expressive in terms of modeling resources, and state updates as well as time relationships in reactive and concurrent systems.

[Hirai, 2000b] used three operators to denote temporal aspects, \( \bigcirc \) (next), \( \square \) (anytime), and \( \Diamond \) (sometime) and recast the meaning of \(!\). Adding \( \bigcirc \) to a formula \( A \), i.e. \( \bigcirc A \), means that \( A \) can be used at the next time point and exactly once. Similarly, \( \square A \) means that \( A \) can be used exactly once and anytime. \( \Diamond A \) means that \( A \) can be used once sometime. Hirai used \(!A \) to express that \( A \) can be used at anytime and for any desired number of times. As a result, the relationships in time among events and actions can be modeled naturally in TLL.

Our work makes use of TLL developed by Hirai in a framework of modeling agent rules for interaction.
Sequent Calculus for Temporal Linear Logic

Apart from the normal sequent calculus rules for linear logic, TLL has extra rules to deal with time. The relevant rules are:

**Anytime - Left**

\[
\frac{\Gamma, A \vdash \Delta}{\Gamma, \Box A \vdash \Delta} (\Box Left)
\]

\(\Box A\) here means that the resource \(A\) is available at a time point of our choice. Therefore, with \(\Box A\), we can choose to consume \(A\) at the appropriate time - now - as in the hypothesis \(\Gamma, A \vdash \Delta\) — so that we can conclude \(\Gamma, \Box A \vdash \Delta\).

**Sometime - Right**

\[
\frac{\Gamma \vdash A}{\Gamma \vdash \Diamond A} (\Diamond Right)
\]

\(\Diamond A\) here means that the resource \(A\) is to be produced at a time point of our choice. We can choose to produce \(A\) now as in the hypothesis \(\Gamma \vdash A, \Delta\) so that we can conclude \(\Gamma \vdash \Diamond A\).

**Anytime - Right**

\[
\frac{!\Gamma, \Box \Pi \vdash A}{!\Gamma, \Box \Pi \vdash \Box A} (\Box Right)
\]

In the context that all resources can be consumed anytime of our choice, if we can derive \(A\) now, we can also derive \(A\) at any time point.

**Sometime - Left**

\[
\frac{!\Gamma, \Box \Pi, A \vdash \Diamond A}{!\Gamma, \Box \Pi, \Diamond A \vdash \Diamond A} (\Diamond Left)
\]

In the context that all other resources can be consumed and produced anytime we choose, we can choose to consume \(A\) now or anytime as required.

**Next time**

\[
\frac{!\Gamma, \Box \Pi, \Xi \vdash A}{!\Gamma, \Box \Pi, \Box \Xi \vdash \Box A} (\Box)
\]

In the context that all other resources can be utilized anytime we choose, if now consuming \(\Xi\) can help produce \(A\), then consuming \(\Pi\) at the next time point can help produce \(A\) at the next time point.

### 2.6.1 Differences of Temporal Operators in TLL

In most temporal logics, “\(\Diamond A\)” means \(A\) is true sometimes, “\(\Box A\)” means that \(A\) is always true, “\(\Diamond A\)” means that \(A\) is true from the next time point, and \(A \cup B\) asserts that \(A\) is true until a time when \(B\) is true. However, as TLL is not particularly concerned with changing of truth values but manipulation of resources over time, the meanings of these temporal...
operators need to be clarified in this new context.

The temporal "next time" "\(\circ\)" still refers to the next time point but has different implications. In temporal logic, the formula after "\(\circ\)" is valid at the next time point and still valid at the subsequent time points whereas in TLL, the formula can only be used at the next time point and is removed when the time passes beyond the next time point.

The discrepancies in meanings of \(\Box\) and \(\Diamond\) between temporal logics and TLL are revealed by examining the TLL sequent calculus rules associated with them.

\(\Box\) preceding a linear logic proposition means that the corresponding resource is available at any time point but exactly only at that chosen time [Hirai, 2000b]. This can be seen as follows. Considering proofs constructed via sequent calculus rules, in all the rules regarding the passage of time, especially the rule \(\circ\), we can see that any formulas preceded by \(\Box\) in the context of the antecedent, for example, \(\Box \Pi\), will survive over the passage of time - appearing as \(\Box \Pi\) in the context of the consequence. This property means that without being used, \(\Box \Pi\) lasts over time.

Regarding the derivation of \(\Box A\), it is governed by the rule:

\[
\frac{\Gamma, \Box \Pi \vdash A, \Delta, ?, \sum} {\Gamma, \Box \Pi \vdash \Box A, \Delta, ?, \sum} \quad \Box \text{right}
\]

which means that \(\Box A\) is resulted from the capability to derive \(A\) at any time (the antecedent). In other words, given this capability, we can choose to utilize \(\Box A\) as \(A\) at any time of our choice.

Together with the continuance of \(\Box A\) over time, this means that if we do not take \(\Box A\) as \(A\) now, we can reserve our choice for the future. Hence, in a model of discrete time as of TLL, \(\Box A\) lasts until the time point it is instantiated into \(A\). This property enables \(\Box\) to reflect the continuance of resource \(A\) over time.

The dual of \(\Box\) is \(\Diamond\). Though \(\Diamond\) also models the continuance of resource \(A\) over time, the difference lies in the rule

\[
\frac{\Gamma \rightarrow A, \Delta} {\Gamma \rightarrow \Diamond A, \Delta} \quad \Diamond \text{right}
\]

which means that \(\Diamond A\) is resulted from a capability to derive \(A\) at a specific time. Hence, \(\Diamond A\) can not be safely instantiated into \(A\) at any other time of our choice.

Therefore, though both \(\Box\) and \(\Diamond\) refer to a point in time, the choice of which time is different. Regarding \(\Box\), the choice is internally decided, as appropriate to one’s own capability. With \(\Diamond\), the choice is external. Strictly speaking, \(\Box\) refers to outer non-determinism and \(\Diamond\)
Temporal Linear Logic

refers to inner non-determinism.

Example

To illustrate the expressiveness of temporal linear logic, we use an example adapted from [Harland and Winikoff, 2004]. We consider a restaurant menu, comments are provided in italic:

chicken soup or seafood soup (in season)- one kind of soup will be served, depending on the availability in the season and we cannot choose which one

main course

chips (all you can eat)- we can eat as much as we want - chips are “unlimited”

tea or coffee- we can choose either one, tea or coffee

as a result of having a meal, if we don’t pay in cash then we pay by check or vice-versa.

Assume that the meal costs us $80.

A representation in linear logic can be derived as follows:

\[
((\text{chicken soup} \oplus \text{seafood soup}) \otimes \text{main} \otimes !\text{chips} \otimes (\text{tea} \& \text{coffee})) \rightarrow (\$80_{\text{cashpay}} \& \$80_{\text{checkpay}})
\]

A consumption of a meal above results in $80 dollars of payment either in cash or by check but not both.

A TTL’s representation includes also temporal operator next \(\circ\). Here, we take into account the time order in serving food. After the soup is the main course, then tea or coffee is provided next. Payment is done after having the meal. Chips can be served anytime during the meal (! operator).

\[
[((\text{chicken soup} \oplus \text{seafood soup}) \otimes \circ \text{main} \otimes !\text{chips} \otimes (\circ \circ \text{tea} \& \circ \circ \text{coffee})) \rightarrow \]
\[
[\circ \circ \circ \circ \$80_{\text{cashpay}} \& \circ \circ \circ \circ \$80_{\text{checkpay}}]
\]

2.6.2 Encoding in Temporal Linear Logic

We consider temporal operators of TLL next \(\circ\), anytime \(\boxdot\) and sometime \(\lozenge\). The meaning of ! is recast to extend the meaning in LL to be applied to all the time points.

Time operator next \(\circ\) refers to the next time point. It can be used to provide a time order or a separation in time of formulas. For example, \(A \otimes \circ B \rightarrow C \otimes \circ \circ D\) is a way to replace \(A\) (A is consumed) now and \(B\) (B will be consumed) at the next time point to get \(C\) (C is produced) now and \(D\) (D will be produced) at the second next time point.
The use of multiple $\Diamond$ allows to express any specific future time point. $\Diamond^n$ is used as a shorthand for $n$ applications of $\Diamond$. $\Diamond^n A$ means $A$ will be available at the $n^{th}$ next time point.

The notion anytime $\Box A$ and its dual sometime $\Diamond A$ both denote that “as long as” resource $A$ is available or action $A$ is done, either at present or in the future. For example $A \otimes \Box B \rightarrow \Diamond C$ means that $A$ is consumed now and it also requires that $B$ is consumed either now or in the future to derive $C$ at the next time point. Though both $\Box$ and $\Diamond$ refer to a point in time, in our interpretations, they exhibit different behaviors when it comes to deciding the actual time point. The choice of which time is internally decided for $\Box$ and externally decided for $\Diamond$.

Combining time operators adds more expressive power. For example,

- at or after a time point $n$, we can make $A$ available anytime: $\Diamond^{n-1} \Box A$.
- at the time of our choice, $A$ becomes available and $B$ becomes available from the next time point: $\Box (A \otimes \Diamond \Box B)$
- $A$ is available once during a specific period ($m$ to $n$) and we can choose the actual time: $\Diamond^m A \& \Diamond^{m+1} A \& \ldots \Diamond^n A$.

### 2.6.3 Comparison between TLL encoding and LL encoding

Using TLL, we can encode the time notion directly onto LL formulas by adding temporal operators - next $\Box$, anytime $\Diamond$ and sometime $\Diamond$. As a result, we can specify the availability of resources and execution of actions over time. These enable agents’ state information to be specified against time points. Moreover, as rules are constructed on TLL formulas, resource utilization via applications of rules or state transitions is also encoded as processes over time. The inclusion of temporal operators in TLL allows a more faithful specification of agent interactions.

TLL allows detection of time constraints. Agents can place time constraints into requests or proposals made to other agents. Using temporal operators, we can

1. specify the constraints in time of resources consumption or production.

2. allow agents to find the appropriate time points so that commitments can be carried out to satisfy those time constraints.
3. allow agents to attempt to construct a time sequence of actions and resource consumption and production.

2.6.4 Relevant Properties in TLL

The properties below clarify the interplay between the LL connectives and temporal operators. The reverse direction does not hold for these properties.

1. $\bigcirc^n A \vdash \lozenge A$; $\bigcirc^n (A \& B) \vdash \bigcirc^n (A \oplus B)$

2. $\bigcirc^n (A \& B) \vdash \bigcirc^n A \& \bigcirc^n B$; $\square (A \& B) \vdash \square A \& \square B$; $\lozenge (A \& B) \vdash \lozenge A \& \lozenge B$

3. $\bigcirc^n A \otimes \bigcirc^n B \vdash \bigcirc^n (A \otimes B)$; $\square A \otimes \square B \vdash \square (A \otimes B)$; $\lozenge A \otimes \lozenge B \vdash \lozenge (A \otimes B)$

4. $\bigcirc^n A \oplus \bigcirc^n B \vdash \bigcirc^n (A \oplus B)$; $\square A \oplus \square B \vdash \square (A \oplus B)$; $\lozenge A \oplus \lozenge B \vdash \lozenge (A \oplus B)$

5. In particular, $\bigcirc (A \otimes B) \not\vdash \bigcirc A \otimes \bigcirc B$;
   $\square (A \otimes B) \not\vdash \square A \otimes \square B$;
   $\lozenge (A \otimes B) \not\vdash \lozenge A \otimes \lozenge B$
   which means that we cannot guarantee to derive the separate existence of $A$ and $B$ at the next time point (respectively anytime, sometime) from the joint existence of $A$ and $B$ at the next time point (resp. anytime, sometime).

TLL has three ways in which formulas can be removed. The first is having the positive formulas being connected with the corresponding negative formulas via the multiplicative conjunction $\otimes$, for example $A \otimes A^\bot \vdash \bot$. The second is via an application of the linear implication $\to$, which removes formulas on its left hand side. The third, which is not in linear logic, is due to passing the formula’s expiration time. This is the case when formulas are available only at a particular time point and hence no longer exist when the time passes that point.

However, there is a trade off between expressiveness and complexity or decidability among fragments. One of the weaknesses of linear logic and hence temporal linear logic is their decidability. The full fragment of linear logic is not decidable [Lincoln, 1992]. This also makes full fragment of TLL not decidable as TLL introduces more operators on top of linear logic.
2.6.5  Modeling Agent Systems using TLL

As in LL, the two sides of a TLL consequence are dual to each other. In the context of resource manipulation among agents, the two sides can be used to model DEMAND (the right side) and SUPPLY (the left side). State formulas of agents can be viewed as occurring on the SUPPLY side while goals formulas can be viewed as occurring on the DEMAND side. As a result of an application of the consequence, formulas on the left side - what is required - are consumed to make formulas on the right side available.

In particular, the views on the determinism of $\&$ and $\oplus$ are different on each side. On the SUPPLY side, if $A \& B$ is available, then we can freely choose $A$ or $B$, i.e. $\&$ is internally determined. If $A \oplus B$ is available, one of them is available but we do not know which, i.e. $\oplus$ is externally determined. On the contrary, on the DEMAND side, if $A \& B$ is demanded, the ability to choose between $A$ and $B$ is also required. Hence, one has to prepare both $A$ and $B$ even though only one will be used, i.e. $\&$ is externally determined. If $A \oplus B$ is on the DEMAND side, as there is no particular requirement for either one, supplying any of $A$ or $B$ is sufficient, i.e. $\oplus$ is internally determined.

Moreover, on the SUPPLY side, $\Box A$ means that we have $A$ to be used exactly once anytime while on the DEMAND side, $\Box A$ requires the ability to provide $A$ anytime as requested. $\Diamond A$ on the SUPPLY side states that we have $A$ available for use exactly once but we do not know at which time. On the DEMAND side, $\Diamond A$ only requires $A$ to be provided at a time of our choice.

Regarding agent reasoning, due to its basis in sequent calculus, TLL can be used for reasoning in a similar manner to LL.

2.7  Timed Petri Nets

2.7.1  Petri Nets

Petri nets (PNs) [Petri, 1966] are graphical and mathematical techniques useful for modeling concurrent systems.

The primitive concepts of Petri nets are places, transitions, directed arcs and tokens. PNs is underpinned by a directed, weighted, bipartite graph whose nodes are either places or transitions and arcs are connecting between places and transitions. Tokens are used to define execution of Petri nets. Each place can potentially hold a non-negative number of tokens. Transitions define how tokens may be transferred from the transitions’ input places to output
Timed Petri Nets

places. Directed arcs connect (input) places to transitions and transitions to (output) places. Directed arcs indicate which input places the tokens come from and which output places they go to for the transitions.

During PNs’ execution, firings of transitions occur and tokens are moved from places to places and the number of tokens in the Petri nets may change. An assignment of tokens to places of a Petri net in which each place is assigned a non-negative number of tokens is called a marking.

Formally, a Petri net is described as a 5-tuple $N = (P,T,I,O,M_o)$ where:

- $P = \{p_1,p_2,p_3,\ldots,p_m\}$ is a finite set of places
- $T = \{t_1,t_2,t_3,\ldots,t_n\}$ is a finite set of transitions, $P \cup T \neq \emptyset$, $P \cap T = \emptyset$
- $I = (P \times T) \rightarrow \mathbb{N}$ is an input function. $I$ specifies directed arcs from places to transitions. $\mathbb{N}$ is a set of non-negative integers.
- $O = (T \times P) \rightarrow \mathbb{N}$ is an output function. $O$ specifies directed arcs from transitions to places.
- $M_o = P \rightarrow \mathbb{N}$ is the initial marking.

Often, weights of arcs correspond to the number of tokens required from the connected input places or the number of tokens that will be deposited in the connected output places.

A transition is enabled if each of its input places is marked with at least the number of tokens corresponding to the weight of the connecting arc. An enabled transition may be fired if the associated event occurs. Firing of an enabled transition removes tokens at the input places and adds tokens to the output places. The exact numbers of tokens removed or added correspond to the weights of the connected arcs. Mathematically, firing of a transition $\tau$ at the marking $M$ yields:

$$M'(p) = M(p) - I(p, \tau) + O(\tau, p)$$

for any place $p$ of the Petri net, where $M(p)$ returns the number of tokens present at the place $p$, $I(p, \tau)$ returns the number of tokens (zero or positive) will be removed as input of the transition from the place $p$, and $O(\tau, p)$ returns the number of tokens (zero or positive) that will be inserted as output of the transition at the place $p$. Moreover, transitions can take place without any input places (source transitions) or without any output places (sink transitions).
We consider a typical example of modeling by Petri nets. Places represent conditions and transitions represent events. The presence of a token in a place indicates that the condition is true. Another view is that the number of tokens present in a place corresponds to the number of data items or resources available. An event is modeled as a transition; input places are then pre-conditions while output places are post-conditions. The initial marking specifies the initial conditions of the modeled system. Changes in markings of the Petri net during an execution then indicate changes in behaviors of the system regarding its states and updates.

We consider a chemical reaction $2H_2 + O_2 = 2H_2O$ [Murata, 1989]. Each molecule can be regarded as a token. It can be seen that two tokens at the place $H_2$ can be combined with a token from the place $O_2$ to form two tokens at the place $H_2O$. The initial marking is assumed to include two tokens at each input place — $H_2$ and $O_2$. A firing of the transition will remove two tokens at the place $H_2$ and one token at $O_2$ and add two tokens to $H_2O$. The resulting marking contains one token at $O_2$ and two tokens at $H_2O$.

By representing (updating) actions and events as transitions, pre-conditions as input places, post-conditions as output places, resources or data items as tokens, we can see that Petri nets can be quite expressive and advantageous regarding modeling many important aspects of systems [Wang, 1998]. This include modeling:

- sequential order of execution or precedence relations as among transitions. Even causal relationships can be modeled in this manner.
- conflicts as between transitions that use shared tokens;
- concurrency as concurrent transitions;
- synchronization as firing of the transition. Tokens may arrive differently at each input places and when there are sufficient tokens to enable the transition, a synchronization among available tokens at different input places takes place as the transition is fired.
- merging as combining various token sources from different places and transitions onto one place;
- confusion as which transitions will take place when two concurrent transitions have their respective inputs both shared by another transition.
- mutual exclusion as transitions that can not take place concurrently or sequentially due to constraints on shared resources.
• priorities as that a transition takes place under a condition which is determined by another transition already being fired. One way to achieve this is to use inhibitor arcs. An inhibitor arc can prevent a transition from being fired if there is a presence of tokens at the input places of the inhibitor arc. Hence, making the required tokens at input places of inhibitor arcs present or absent can turn on or off the firing of some transitions permanently or before or after some other transitions, hence expressing priorities among transitions.

There are a number of advantages that PNs bring to modeling systems. Visual modeling allows PNs to add comprehension and unambiguity to communication of models among users, such as engineers and customers. With the help of computer tools, interactive graphical simulation of PNs can be produced, which can be quite advantageous in developing complex engineering systems. Having a firm mathematical basis permits Petri net models to be expressed as sets of linear algebraic equations or other mathematical models. This in turn makes PNs suitable for formal analyses such as checking for correct synchronization, freeness from deadlock, true concurrency, and mutual exclusion of shared resources. In fact, properties of Petri nets can be mapped on to the context of the modeled systems to reveal if some functional properties of the system are present [Wang, 1998]. Such important properties include reachability, boundedness, conservativeness and liveness.

2.7.2 Timed Petri Nets

One of the essential limitations of Petri nets is a lack of consideration of duration. To overcome this, PNs have been extended to cover time aspects. In particular, each place, transition or directed arc can be associated with a deterministic firing time or time interval and the resulting PNs are deterministic timed Petri nets. In another form, each transition of PNs is associated with a somewhat random firing time, which corresponds to stochastic timed Petri nets. As a result of such extensions, such time extended PNs or timed Petri nets (TPNs) offer a full treatment of discrete event systems, regarding design, modeling and performance analysis.

Timed Petri nets ([Merlin and Farber, 1976]) are a particular variation of this extension to Petri nets in which there is a specification of two time values for each transition [Wang, 1998]. The first value defines the minimum time that a transition must wait for after it is enabled and before it is fired. The second time value specifies the maximum time for waiting of the transition before firing if it is still enabled. These two time values are taken to be
relative to the first enabling time of the transition. Such extension is likely to be suitable for dealing with constraints on firing durations.

2.7.3 Applications of (timed) Petri Nets

PNs have been extended to modeling a variety of information systems whose characteristics may be being concurrent, asynchronous, distributed, parallel, nondeterministic, and/or stochastic [Murata, 1989]. For example, PNs serve as a popular tool for flexible and automated manufacturing systems [Moore and Gupta, 1996], for performance evaluation [Rama­moorthy and Ho, 1990], for communication networks [Gressier, 1985; Zhu and Denton, 1988], for traffic control systems [Mandroili et al., 1996], and for military and control systems [Wang and Zhou, 1997]. Indeed, Petri nets can represent finite state machines, data flow computation, and synchronization control among multiprocessors sharing the same information or resources, and readers-writers & producers-consumers problems [Murata, 1989].

Variations of time-extensions of Petri nets have also gained popularity in modeling real-time concurrent systems. For example, timed Petri nets have been applied to the modeling and verification of such systems [Bucci and Vicario, 1995; Deng et al., 1998; Tsai et al., 1995]. Moreover, such real-time systems as the Command and Control System (C2 System) used in battle fields is also modeled and has been analyzed by timed extensions of Petri nets. C2 Systems involve a vast and complex integration of multiple techniques and hardware. The performance of C2 Systems is based on measuring the time delay in execution of their generic functions such as target detection, target tracking, and discrimination among targets [Wang, 1998]. In particular, the performance of C2 Systems depends on how well information variables can be reflected in real-time properties of quickly changing physical entities like targets and weapons. Timed Petri nets were used for modeling and performance analysis for a generic naval C2 (command and control) battle group system [Choi and Kuo, 1988] and to evaluate time-related performance of decision making organizations when regarded as an asynchronous concurrent system [Hillion, 1986]. Stochastic Petri nets with exponential firing time transitions were also used to address the information processing capacity of a C2 system [Wang and Zhou, 1997].

2.8 Summary

The chapter discussed relevant work in the areas of flexible protocols, commitments, temporal logic, linear logic, temporal linear logic and (timed) petri nets. On the one hand,
commitment-based approaches explore the benefits of using the notion of commitment in the context of agent interaction, especially making interaction more flexible by structuring protocols around commitments. On the other hand, much existing work in linear logic and temporal linear logic addresses the modeling of exchanging and utilizing resources which is an important and practical aspect of agent interaction. Interesting issues remain as how combining the commitment-based approaches with resource-conscious modeling approaches can benefit the study of agent interaction, especially making it natural to address practical issues of resource exchange and utilization as well as making it more flexible. The next chapters will discuss our modeling, specification and execution frameworks for agent interaction. Also, the links between our frameworks and the popular modeling tool - timed Petri nets - will be discussed.
Chapter 3

Modeling Agent Interaction

As discussed above, traditional approaches to the specification of protocols are too rigid for agent systems. Two criteria for flexibility in agent protocols [Yolum and Singh, 2002b] that we focus on are preserving the agents’ autonomy in their interactive behaviors and allowing agents to explore opportunities and handle exceptions in the face of changes.

In order to model interaction among agents, we make use of the commitment concept, which includes both internal and social commitments. The concept of social commitment has been used as a basis for agent interactions [Singh, 1996] and used to develop execution mechanisms such as commitment machines [Yolum and Singh, 2002a]. Our approach also uses the concept of commitment to capture the meaning of interaction goals, states and interactive actions, and to allow agents to perform mean-ends reasoning over them.

The underlying logic in our declarative approach to protocol specification is temporal linear logic (TLL). As discussed in Section 2.6, TLL has advantages in modeling resources and updates in time. In this chapter, we use TLL to model interaction among agents. In particular, we show how concepts like resources, actions, capabilities, and commitments can be expressed as TLL formulas. We also discuss the roles of these concepts in agent interaction, especially commitments.

The structure of the chapter is as follows. Section 3.1 describes a running example of agent interaction — Cricket Bat Sales. Section 3.2 discusses how TLL formulas can be used to model various concepts in resource based interaction. Section 3.3 discusses the modeling of various forms of commitments and their roles in agent interaction.
3.1 Cricket Bat Sale Example

In this example, the interaction scenarios take place between a merchant agent named Mer, a customer agent named Cus, and a bank agent named Ebank.

Mer has 200 junior cricket bats and 300 heavy cricket bats available for sale. Cus has $50 that can be used to pay via Paypal or by credit card. Cus has a goal of obtaining from Mer two junior cricket bats, and a gift at some time.

Cus initiates the interaction by making a request to Mer. Mer can choose the time to make a sale to meet Cus’s goals. By default, Mer charges $10 per junior cricket bat with no gift. Also, Mer can offer a gift for every purchase of two junior cricket bats. Cus’s address is required for delivery. There are two options for payment, either Cus pays via Paypal or Mer charges by credit card.

If Cus prefers credit payment, Mer needs Ebank to help with charging Cus. There are two possibilities. If Cus’s credit is approved, Ebank will arrange the credit payment. Otherwise, Ebank will indicate to Mer that Cus’s credit is not approved. As a result, Cus may then take the option to pay via Paypal. The interaction ends when goods are delivered, payment is arranged and Cus receives the gift.

The interaction involves exchanges of resources and information among the participating agents Cus, Mer and Ebank. The resources of Mer include junior cricket bats and heavy cricket bats. Cus has money and relevant information such as his address and credit card number. Ebank has the knowledge of how to determine whether a particular credit card transaction is approved or not. There are exchanges of resources such as a junior cricket bat for money and transfers of information such as sending the customer’s credit card number to the merchant, and sending the result of the bank’s decision on a credit card payment to the merchant.

As these agents interact, they explore the capabilities of each other. The capabilities of Mer are, for example, those that allow it to make a sale of cricket bats given the payment is properly arranged. Ebank has the capability to determine whether to approve a payment or not.

There are also various choices that are involved in the interaction and hence alter the interaction sequence. These include the choice of the time Mer does the sale, the sale option Mer chooses to offer Cus, which payment method Cus uses, and, in case of a credit card payment, whether Cus’s credit check is approved.

A key aspect of the flexible approach is that we do not want unnecessary constraints on
the order of actions such as sending address, making payment, and sending junior cricket bats.

We then look at the modeling of the consumption/production aspect of resources and updating of information. Other concepts like actions, capabilities, choices are also examined.

3.2 Resource Based Interaction

Resources are usually used to produce other resources. For example, brewed coffee, milk and sugar are raw materials used to produce a cappuccino. Resources can also be relocated from one place to another and have their ownership changed, such as when a cup of cappuccino is purchased. In the context of agent interaction, agents usually exchange their resources and explore each other’s ability to transform resources to achieve their respective goals. We discuss how temporal linear logic might be used to naturally model the use (consumption) of resources and provide a mechanism for handling resource utilization.

3.2.1 Resources

Using linear logic is a natural way to model the consumption and transformation of resources, as has been discussed in section 2.5. In this section, we focus on using of linear logic for agent interaction.

Each agent possesses a set of resources which can be consumed, produced or transformed into other resources or exchanged with other agents’ resources. As linear logic propositions can be produced, consumed exactly once or transformed from or into other propositions’ formulas, they are quite suitable for modeling resources. We model each unit of resource as a linear logic proposition. It is important to stress that what we are concerned with is the use of resources in agent interaction but not the physical resources themselves. As a result, we are not concerned with the resources’ persistence over time and our modeling does not keep track of the physical existence of resources but only their use.

For example, let $\text{cof\_cup}$ represent a cup of coffee. An introduction of the formula means a cup of coffee is made, while removing it means the cup of coffee is consumed, and $\$5 \rightarrow \text{cof\_cup}$ denotes an exchange of a resource of five dollars for a cup of coffee at a coffee shop.

Quantity of resources is addressed by the corresponding number of linear logic propositions joined together via the $\otimes$ connective. As a shorthand, we denote $\text{cof\_cup} \otimes \text{cof\_cup} \otimes \text{cof\_cup}$ as $3 \text{cof\_cup}$.

The use of next time operator $\bigcirc$ allows us to model when the resource is available. For
example, denoting a junior cricket bat \((jr\_cb)\) which will be available at the second next time point is by \(\bigcirc \bigcirc jr\_cb\). We use a short hand \(\bigcirc^n A\) for \(\bigcirc \bigcirc \ldots \bigcirc A\), where there are \(n\) number of \(\bigcirc\)s.

As the \(\bigcirc\) operator is only associated with a specific time point (i.e. not a duration), the corresponding resource it addresses is not available after that time point, even if it is not used. The operator \(\Box\) is used to overcome this by addressing that the use of the formula can be anytime and if the formula is not used now, then it can be used later. Like \(\bigcirc\), \(\Box A\) also means that the corresponding resource \(A\) is available for exactly one use [Hirai, 2000b]. Because we do not model the persistence over time of physical resources, \(\Box\) is used to with the intended meaning that the resource can be used any time, and not that the resource lasts for a long time.

The interpretation of \(\Box\) in temporal logic corresponds to “always” which makes an assertion true at all future moments [Emerson, 1990]. Our interpretation, in which \(\Box\) makes a formula available at any time of choice but exactly once, is hence different.

The dual of \(\Box\) is \(\Diamond\). \(\Diamond A\) means that \(A\) is available for use but at a time that we do not know. For example, consider a customer who has ordered a pizza by phone. After two time units, the pizza is made and ready for the customer to pick up, which can occur at any time from then on. This is expressed as \(\bigcirc^2 \Box pizza\). From the restaurant’s viewpoint, it does not know the exact time that the customer will collect the pizza, i.e. the restaurant sees this resource as \(\bigcirc^2 \Diamond pizza\).

The presence of a \(\mathcal{LL}\) formula means the encoded resource is available for use. However, in \(\mathcal{TLL}\), the presence of a \(\mathcal{TLL}\) formula means that there is a resource whose availability also depends on the time operators. For example, \(\bigcirc^2 A\) means that the resource \(A\) is available at the second next time point, but not at any other time.

Moreover, given the ability to describe the time points associated with resources’ availability by using \(\mathcal{TLL}\) formulas, changes in resource availability can also be described with precise time points. Such changes include resource consumption, production, relocation, change of ownership and expiration. Indeed, resource consumption is regarded as removing the availability of a resource at a particular time by consuming its formula at that time. Resource production is modeled by creating a resource formula representing the availability of the resource and associating it with temporal operators. Resource transformation can be regarded as consumption of resources followed by production of (other) resources. Resource relocation is changing its location but not its availability. Similarly, a change of ownership of a resource means a change in its owner but not in its availability. Resource relocation and
ownership change are discussed further in the next section.

**Resource Location and Ownership**

In practical systems, resources generally have to be exchanged between agents in order to achieve their goals. In these exchanges, some resources are relocated and/or have their ownership changed. Therefore, it is important to address such changes in location and ownership of resources. We discuss how TLL can be used to naturally to model the updating of location and ownership.

We use a special marker @ to indicate location of resources. Resource A located at agent $\alpha$ is denoted as $A@\alpha$. Changes in the names of the agents where the resource is located then reflect the changes in locations of the resources. Relocation of resources can be modeled by using $\rightarrow$ to signal a transition from the formula representing the resource at its old location to the formula representing the resource at its new location. For instance, if resource A is moved from agent $\alpha$ to agent $\beta$, the updating process is modeled as $A@\alpha \rightarrow A@\beta$. The updating process applies to $A@\alpha$ hence results in $A@\beta$.

Moreover, to recognize the ownership of a resource, we include a footnote to indicate the owner agent. For example, a cricket bat owned by the merchant M is denoted as $crik.b_M$. The footnote, together with the resource formula, can be similarly expressed in logic as $crik.b(M)$ or as a proposition $crik.b_M$. To change ownership of a resource, the updating process makes use of linear implication. For example, when a cricket bat is purchased, ownership passes from the merchant to the customer. The updating is modeled as $crik.b_M \rightarrow crik.b_C$.

Combining the notation for both location and ownership, we have the form $A@\alpha\beta$, which indicates there is a resource A at agent $\alpha$ and owned by agent $\beta$.

The updating of information about location and ownership can be illustrated in an example of buying a cup of coffee. A customer walks into a coffee shop to buy a cappuccino which costs 4 dollars. Items of concern are:

- $4@C_c$: four dollars is currently located at and owned by the customer
- $capu@shop_c$: a cappuccino is located at the shop but owned by the customer. This corresponds to a ticket or receipt handed to the customer who will receive the cappuccino when it is ready.
- $capu@C_c$: a cappuccino is located at and owned by the customer and hence is available for drinking
The purchase of a cappuccino can be regarded as a two phase process. The first one is governed by a rule for buying a cappuccino, which is \(4@C_C \rightarrow O^3 capu@shop_C\). The rule states that if the customer pays 4 dollars (providing \(4@C_C\)), he will receive a ticket/receipt which says that a cappuccino will be available for pick up in three time units \((O^3 capu@shop_C)\). This is depicted in logic as follows

\[
4@C_C \otimes (4@C_C \rightarrow O^3 capu@shop_C) \vdash O^3 capu@shop_C
\]

The second phase is at the third next time point, the customer hands in the ticket/receipt and gets the cappuccino \((capu@C_C)\) according to the rule \(capu@shop_C \rightarrow capu@C_C\). Now the cappuccino is at the customer and he can drink it.

\[
capu@shop_C \otimes (capu@shop_C \rightarrow capu@C_C) \vdash capu@C_C
\]

In this example, the ownership of the cappuccino is appropriately recognized at the time of payment: the customer pays and hence owns the cappuccino. Also, a distinction is made between where the cappuccino is located and who (currently) owns it. The packaging of information about location and ownership into the resource representation hence allows us to faithfully model the process of resource exchanges.

### 3.2.2 Status Information

As mentioned above, we model the use of resources, rather than their physical state. As information is retained after accesses, such modeling of information is troublesome because the formula representing information is removed after use. One way to fix this is to make use of \(!\) to indicate an unlimited number of accesses to the information (anytime). For example, consider a credit card number encoded as \(!cred_no\). This approach, however, is limited. In particular, the information of concern is about the knowledge of the credit card number, which is a dynamic entity and hence is subject to changes. Modeling temporary information using \(!\) is problematic because it is impossible to remove all instances of the formula and hence impossible to update the information.

Another approach is to model information by providing sufficient copies of the formula for use throughout the information’s lifetime. For instance, ”customer has not paid” is encoded as \(\eta \sqcap not_paid\), where \(\eta\) is a chosen number which is large enough to the context.
In this approach, there are a number of ways to update the information. One is to remove all the instances of the formulas, which indicates that the resource has been exhausted. This, however, may cause confusion with the state of no information. Another way is to add a number of formulas of the new value after removing formulas of the old value. This method, however, requires keeping track of the number of formulas and maintaining the mutual exclusive occurrences of formulas corresponding to opposite values.

Our approach, however, keeps information as a single proposition with temporal operators. We ensure that information is not lost by maintaining control over the way it is accessed or updated. In particular, we model access to a piece of information as a use of the formula, followed by a creation of another copy of the formula. This allows us to express both the use of the information and that it persists. Control over the use of information is expressed by using linear implication. For example, accessing the customer’s credit card number is modeled as cred_no → cred_no. Accessing information can be ”bundled” with other processes, such as

\[ cred_no \otimes sale_quote \rightarrow cred_no \otimes sale_quote \otimes (cred_paym \oplus cred_disappr) \]

which expresses a process of credit payment, taking information about the credit card number and a quotation to determine if a credit payment is approved or not. Note that this is dependent on the checking process, and hence is not an internal choice to the agent.

The updating of information is also modeled by the use of linear implication →. For example, the process not_paid → has_paid updates the payment status of the customer by replacing the formula not_paid by has_paid.

Moreover, we use the ownership notation (subscript) to mark the source of information. For example, as the information about the credit card number of the customer comes from the customer, this is expressed as cred_no_C. When a piece of information is brought to an agent, we assume that the agent becomes aware of it. For example, to indicate that the merchant knows the customer’s credit card number, we use cred_no@M. Hence, information exchange, such as when agents need updates from other agents, can be modeled by the relocation of information. Passing information from agent α to agent β is done by replacing the location of the information. For instance, \( cred_no@C_C \rightarrow cred_no@C_C \otimes cred_no@M_C \) expresses sending a customer’s credit card number to the merchant.
3.2.3 Capabilities and Actions

We model agents’ capabilities to produce, consume, relocate and change ownership of resources by declaring the effects they have. In particular, the conditions before and after an application of the capability are specified. The ability to transform from pre-conditions to post-conditions is captured by linear implication $\rightarrow$.

The general form for capabilities is $time_{op}[\Gamma \rightarrow \Delta]$ in which $\Gamma$ is the pre-conditions and $\Delta$ is the post-conditions. $\Gamma$ and $\Delta$ may contain any TLL formulas. $time_{op}$ indicates the time when the capability is available and can be applied.

Any resources in the pre-conditions of a capability are consumed and as a result, resources in the post-conditions are produced. This can be thought as a resource transformation process. Resource relocations, as discussed in previous section 3.2.1 are realized by keeping the respective resource and changing the names of the agents where those resources are located. Similarly, changes in ownership of resources are modeled by keeping the resources and changing the names of the owners.

To illustrate a capability to consume and produce resources, we consider a resource transformation process that produces a cappuccino. A cappuccino is made of sugar, milk and brewed coffee. This is modeled as $brewed_{coffee} \otimes milk \otimes sugar \rightarrow capu$. The pre-conditions contain some units of brewed coffee, milk and sugar. When they are provided, the capability can be applied. An application of the capability removes all the formulas of brewed coffee, milk and sugar and produces a formula of the cappuccino, which reflects the transformation process.

An example of resource exchange is selling cricket bats. The formula below

$$20@C \otimes 2\text{crick}_b@M \rightarrow 20@M \otimes 2\text{crick}_b@C$$

encodes a capability of a merchant (M) to sell 2 cricket bats for 20 dollars to customers (C). As can be seen, the 20 dollars is relocated from the customer to the merchant and the cricket bat is relocated from the merchant to the customer. Ownership of those resources is also changed accordingly.

Moreover, the operator ! is also suitable to express those capabilities that can be applied an infinite number of times. Their formulas are preceded by !. What then constrains the application of these capabilities is the limited availability of the resources of the pre-conditions.

Application times of a capability are bounded by the time constraints described by its time operator. If the time operator is $\bigcirc^n, n > 0$ then the rule can not be applied after $n$
time points. If the time operator is \( \Box \) then the application times of the rule can be anytime (but may be constrained by other factors). If the time operator is \( \Diamond \) then the application time of the rule will be sometime that the agent does not know. The operator \( ! \) does not impose any constraint on time or the number of rule applications.

Application times of capabilities depend on a number of factors. Firstly, for a capability \( \text{time}_\text{op} [\Gamma \rightarrow A] \) to be applied, it is necessary that the pre-condition \( \text{time}_\text{op} \Gamma \) is provided so that \( \text{time}_\text{op} [\Gamma \otimes (\Gamma \rightarrow A)] \vdash \text{time}_\text{op} A \). We can perform the capability now as in

\[ \Box^n \Gamma \otimes \Box^n [\Gamma \rightarrow A] \vdash \Box^n A, \]

or at \( \Box^n \) as in

\[ \Box^n [\Gamma \otimes (\Gamma \rightarrow A)] \vdash \Box^n A \]

or at some time in between, \( \Box^m \), as in

\[ \Box^m [\Box^{n-m} \Gamma \otimes \Box^{n-m} (\Gamma \rightarrow A)] \vdash \Box^n A. \]

The application time can be quite flexible, at any time between now and \( \Box^n \). Lastly, if pre-condition formulas and the capability formula are available over a time range instead of at a specific time, the application time of the capability can be chosen such that the post-conditions are available at the most desirable time.

Note that capabilities and resources are modeled as separate and independent formulas. Creation or applications of capabilities are automatically reflected in the state by the presence or removal of their corresponding formulas. Therefore, the loss of a resource does not affect the capabilities (but may reduce the chance of having enough resources to apply the capability). The capabilities either exist or are removed as a result of their application, and have no partial updates.

Furthermore, actions for which we do not wish to model any changes in resources or system properties as a result of their application can be modeled as TLL propositions in a similar manner as modeling resources. In particular, a formula representing the action denotes that the action is available. When the action is used, its formula is also removed. For instance, the action of blending (\( \text{blend} \)) can be added to the pre-conditions of the capability of making cappuccino: \( \text{brewed}_\text{coffee} \otimes \text{milk} \otimes \text{sugar} \otimes \text{blend} \rightarrow \text{capu} \). In this example, removal of the pre-conditions’ formulas means that the raw materials are consumed and the action blending is carried out.
Actions of agents that involve resource changes or changing system properties can be represented in a similar manner to that of capabilities by connecting their pre-conditions with their post-conditions via linear implication. As linear implication involves a definite transition, fulfilling the pre-conditions always leads to the post-conditions, which means the action is successful. To model actions that can fail, the post-conditions can be modeled with the use of an internal choice or an indeterminate possibility. Specifically, to model an action (to produce B) that has pre-conditions A and post-conditions B, instead of modeling as \( A \rightarrow B \), we could use \( A \rightarrow (1 \oplus B) \) or \( A \rightarrow (1 \& B) \). Note that 1 means nothing has been derived. The post-condition \( 1 \oplus B \) means that the outcome of the action is possibly 1 (failure) or B (success) and the agent cannot control which. Hence the outcome \( 1 \oplus B \) means that the action (to produce B) can possibly fail. On the other hand, the post-condition \( 1 \& B \) means that the action can fail or succeed, depending on our choice.

It can be seen that with such modeling, the pre-conditions are always consumed whether the action fails or not. To model actions which do not consume their pre-conditions on failure, we could use such modeling as \( A \rightarrow (A \oplus B) \) (or \( A \rightarrow (A \& B) \)), where if \( A \) results, the action (to produce B) fails and the pre-conditions are fully restored.

Given that we are mainly concerned with the modeling of interactions and their protocols, to simplify the handling of actions, we assume that actions of agents are always successful, bearing in mind that it is possible to cater for failures of actions (as mentioned above).

The modeling of actions (with and without using linear implication) treats execution time of an action as a single time point. The modeling is suitable to address the declaration aspect of actions or capabilities, which is important in agent planning. For actions whose execution is over a period of time, TLL does not have any temporal operator that naturally models a duration. For actions modeled without pre-conditions and post-conditions, we model such actions that have continuous execution by spreading copies of the action’s formula over these time points to simulate a continuance in such discrete time system like TLL. For example, in the coffee making example, we can describe the action of blending over three time points as \( \textit{blend} \otimes \textit{blend} \otimes \textit{blend} \otimes \textit{blend} \). The use of all these formulas means repeating executing the action blending over three consecutive time points, which simulates continuous execution over the period.

For actions that have pre-conditions and post-conditions, we can model the action of these as a series of sub-actions, with each having a separate set of pre- and post- conditions. These sub-actions can in turn be modeled as actions whose execution time is at a single time point.
Furthermore, given actions that involve resource or changing system properties are represented in the same way as capabilities, for convenience, we refer to them as a form of capabilities. We then refer to actions as only those that we do not wish to model their effects as changes in resources or system properties.

### 3.2.4 Internal Choices and Indeterminate Possibilities

As discussed in section 2.5, internal choices and external choices can be modeled using the connectives \& and \oplus in linear logic, and hence TLL. Our approach to modeling internal choices and indeterminate possibilities is similar in that internal choices are modeled by using \& and indeterminate possibilities are modeled using \oplus. Indeterminate possibilities here refer to the possibilities that the agents do not know their actual outcomes (which possibilities will turn out to be true). We will further discuss the modeling and strategies in dealing with internal choices and indeterminate possibilities in Chapter 5.

### 3.2.5 Cricket Bat Sale Example

Various concepts of resource, action, capability, goal, choice and information in the cricket bat sale example (section 3.1) are modeled as follows.

**At agent Mer**

Mer has 200 junior cricket bats and 300 heavy cricket bats available for sale. The availability of these cricket bats is anytime and hence encoded with \(\Box\). The shorthand for 200 junior cricket bat formulas that are “and”ed \(\otimes\) together is 200\(\Box \ jr_{cb}\). All are at Mer and owned by Mer: 200\(\Box \ jr_{cb}@M_{M}\). The encoding of 300 heavy cricket bats is similar.

\[200\Box \ jr_{cb}@M_{M} \otimes 300\Box \ heavy_{cb}@M_{M}\]

Mer can perform the action of issuing a quotation anytime and can repeat this action a number of times: \(\eta \Box \ issue\_quote@M_{M}\).

We consider a number of capabilities of Mer.

**Capability 1.** Mer has a sale capability in that it sells junior cricket bats for 10 dollars each.

Pre-conditions of the capability include payment by the customer and having an address for delivery \(\text{addr}@C_{c}\) as well as an available junior cricket bat at the merchant \(\text{jr}_{cb}@M_{M}\). Payment by the customer is arranged by having 10 dollars available \(10\text{$}@C_{c}\) and being carried out via a payment method — by credit card \(\text{by\_cred}@M_{c}\) or via Paypal \(\text{via\_PP}@M_{c}\).
The choice of payment method is the customer’s choice \((\text{via } PP@M_C \oplus by\_cred@M_C)\) and Mer accepts either of them.

The pre-conditions of the capability are expressed as:

\[
(10\$@C_C \otimes (\text{via } PP@M_C \oplus by\_cred@M_C) \otimes addr@C_C \otimes jr\_cb@M_M)
\]

Post-conditions of the capability specify that the customer’s address is received at the merchant (\(\square addr@M_C\)), a junior cricket bat is received by the customer (\(\square jr\_cb@C_C\)), and the money is received by the merchant (\(\square 10\$@M_M\)). These post-conditions reflect the results of applying the sale capability to the pre-conditions. The post-conditions are expressed as:

\[
10 \square \$@M_M \otimes \square jr\_cb@C_C \otimes \square addr@M_C
\]

As the merchant can sell anytime, the capability is preceded by the \(\square\) operator. Also, the capability can be applied \(\eta\) number of times, where \(\eta\) is big enough in the context so that the merchant is capable of selling any amount of stock available.

Combining all elements, the sale capability of the merchant is expressed as

\[
\eta \ square [10\$@C_C \otimes (\text{via } PP@M_C \oplus by\_cred@M_C) \otimes addr@C_C \otimes jr\_cb@M_M]
\]

\(\rightarrow 10 \ square \$@M_M \otimes \square jr\_cb@C_C \otimes \square addr@M_C]\]

**Capability 2.** The merchant is also capable of making a sale of two junior bats at once plus a gift offer. Modeling this capability is similar, with an addition of a gift from the merchant (\(gift@M_M\)) in the pre-conditions and the gift being received by the customer in the post-conditions (\(gift@C_C\)). The capability is represented as follows. Note that the merchant is capable of making at most 20 sales of this kind.

\[
20 \ square [(20\$@C_C \otimes (\text{via } PP@M_C \oplus by\_cred@M_C) \otimes addr@C_C) \otimes 2jr\_cb@M_M \otimes gift@M_M]
\]

\(\rightarrow (20 \ square \$@M_M \otimes \square 2jr\_cb@C_C \otimes \square gift@C_C \otimes \square addr@M_C)]\]

**Capability 3.** Mer has the capability to accept payment by credit card which is performed by giving EBank Cus’s credit number and getting EBank to arrange a credit payment
to Mer. The action of sending Cus’s credit card number to EBAnk is provided in the capability and hence given a negative form on the left hand side of \(\neg\) (similar issues arise with Capability 5, as discussed below).

\[
\eta \quad \Box \left[ (\text{cred}_n\neg@M_C \rightarrow \text{cred}_n\neg@B_C) \downarrow \otimes \text{cred}_\text{paym}@M_B \rightarrow \neg \Box \text{cred}_\text{paym}@M_M \right]
\]

**Capability 4.** Mer is capable of giving EBAnk a quotation (\(\text{quote}@B_M\)) given a request for a quotation (\(\text{quote}_r\text{eq}@M_B\)) from EBAnk. When this capability is performed, an action of issuing a quote (\(\text{issue}_\text{quote}@M_M\)) is also carried out.

\[
\eta \quad \Box \left[ \text{issue}_\text{quote}@M_M \otimes \text{quote}_r\text{eq}@M_B \rightarrow \neg \Box \text{quote}@B_M \right]
\]

**At agent EBAnk**

**Capability 5.** EBAnk is capable of checking a customer’s credit rating and arranging a credit payment. This capability requires the credit card number of the customer, and a quotation from the merchant. A request for a quotation is also given to Mer in this capability. The outcome of performing the capability includes an arrangement of the credit payment or an indication of failure of the customer credit check. The choice here depends on EBAnk’s internal processing.

\[
\eta \quad \Box \left[ \text{quote}@B_M \otimes \text{cred}_n\neg@B_C \otimes \text{quote}_r\text{eq}@M_B \downarrow \rightarrow \neg \Box \text{cred}_\text{paym}@M_B \oplus \Box \text{cred}_\text{disapp}@M_B \right]
\]

Note that on the left hand side of \(\neg\) in the capability, \(\text{quote}_r\text{eq}@M_B\) is given in its negated form \(\text{quote}_r\text{eq}@M_B^\perp\). Because the left hand side of \(\neg\) is the pre-conditions — what is required — the negation of \(\text{quote}_r\text{eq}@M_B\) then indicates a provision. This can be seen clearly in the following proof steps (abstract). Consider a capability \(A^\perp \rightarrow B\) in the following context of using \(\Gamma, \Gamma'\) to derive \(\vdash \Delta, \Delta':\)

\[
\begin{align*}
\Gamma, A &\vdash \Delta \\
\Gamma' &\vdash A^\perp, \Delta \quad \dashv R \\
&\Gamma', B \vdash \Delta' \\
\Gamma, \Gamma', A^\perp &\rightarrow B \vdash \Delta, \Delta' \rightarrow L
\end{align*}
\]

It can be seen that \(A^\perp\) appears in the pre-conditions of the capability \(A^\perp \rightarrow B\) means a provision of \(A\) (as in \(\Gamma, A \vdash \Delta\)) to what have been required (\(\Gamma, \Gamma'\)). An application of the capability \(A^\perp \rightarrow B\) provides both \(A\) and \(B\) to what is currently available.

**At agent Cus**

Cus has a resource of 50 dollars available for use anytime, information about his credit card number and an address:
Commitment based Interaction

\[ \square \text{\$50@C}_c \otimes \eta \quad \square \text{addr}@C_c \otimes \eta \quad \square \text{cred_no}@C_c \]

Cus also can perform an action of paying via Paypal: \( \eta \quad \square \text{via}_\text{PP}@C_c \).

**Capability 6.** Cus has a capability of paying by credit card. This capability includes an action of providing Mer with the credit card number

\[ ((\text{cred_no}@C_c \rightarrow \text{cred_no}@M_c)^\perp) \]

This action is given in its negation form and as explained earlier, negation form on the left hand side of \( \rightarrow \) means a supply rather than a demand. The capability also requires Mer to arrange the credit payment \( (\text{cred_paym}@M_m) \).

\[ \eta \quad \square [((\text{cred_no}@C_c \rightarrow \text{cred_no}@M_c)^\perp \otimes \text{cred_paym}@M_m \rightarrow \text{by_cred}@M_c)] \]

Agents interact to exchange resources and explore each other’s capabilities. The next question is how to coordinate agent interactive actions and provide agents with a flexibility over their actions. The following section discusses the use of commitment to achieve flexibility in agent interaction.

### 3.3 Commitment based Interaction

The notion of social commitment is an important mechanism to agent coordination. Commitments are persistent. As discussed in Section 2.3.2, they provide certain level of stability and predictability, which allows agents to coordinate their activities with others.

Furthermore, the notion of social commitment captures the meaning of interactive actions and hence protocols specified in terms of commitments provide agents with more flexibility over their interactive behaviors. Our approach makes use of the concept of social commitment as a means to construct agent interaction.

Commitments are pledges or promises [Jennings, 1993] such as that some resources will be made available or that some actions will be carried out by the agent.

Commitments have been modeled as abstract data types [Yolum and Singh, 2002a; Chopra and Singh, 2004; Mallya et al., 2003; Yolum and Singh, 2002b] or as objects of an abstract (commitment) class [Fornara and Colombetti, 2002]. However, we try to take advantage of properties of the TLL to model commitments as dual to the resources and actions required to fulfill them.
Commitment based Interaction

We discuss how various types of commitments are modeled and handled and also their relationships to resources, actions, and capabilities.

3.3.1 Internal Commitments and Social Commitments

Internal commitments are commitments by an agent to itself. We do not address the notion of internal commitment in relation to mental states of agents but refer to the scope of commitment. Internal commitments can be regarded as matters private to the agent and hence information about them is not intended for other agents. Social commitments are commitments made by one agent to another. Internal commitments are different from social ones in that social commitments are, in principle, made known to other agent(s) in the interaction. Being known to other agent(s) is essential for social commitments to provide a certain level of predictability.

We take the view that the distinction between social and internal commitments is primarily based on the context in which they occur. If the context is within a single agent, the commitments refer to internal commitments. If the context is an interaction with other agent(s), the commitments are social. In our modeling, we do not make a distinction between representations of internal and social commitments. We assume that different treatments of them are determined appropriately by the context in which they appear. This assumption allows us to simplify the representation of commitments by not explicitly specifying the agents involved in the commitment.

Both social commitments and internal commitments can be base commitments and conditional commitments which are described in the sections 3.3.2 and 3.3.4.

Base commitments are interpreted within the context of conditional commitments agreed upon by the interacting agents. Specifically, in a conditional commitment $\Gamma \rightarrow \Delta$ from agent $\alpha$ to agent $\beta$, the base commitments $\Gamma$ and $\Delta$ are the social commitments of agent $\alpha$ and agent $\beta$ respectively. An example of such social commitments is a social commitment from a merchant to give a cricket bat to a customer who pays.

Our approach focuses on modeling commitments as dual to the resources & actions required to fulfill them; in other words, we focus on the handling of commitments. As mentioned above, we do not differentiate between internal and social commitments, nor do we keep track of to whom the commitment is made. It is important to know this when deliberating whether to fulfill a commitment or not, or to analyze what commitments have been broken. However, such issues are outside the scope of this thesis (although our approach does
not exclude the possibility of keeping track of the agent to whom a commitment is made).

### 3.3.2 Base commitments

Non-conditional commitments are referred to as base commitments. For example, in the cricket bat sale example, once the customer has paid, the merchant has a base commitment of sending the customer two cricket bats. We consider modeling base commitments as dual to what is required to fulfill them by exploring the duality between positive and negative formulas. If we consider positive formulas as what is already provided, due to the duality of negation, their negations can be treated as of what is demanded. In this context, base commitments are formulas in demand and hence can be modeled by negative formulas.

In particular, we assume that agents have a common overall goal to satisfy all the demands, which means removing formulas in negated form. Achieving this goal then exerts a driving force over the manipulation of agents’ formulas to arrive at a state of no negative formulas. This force can be used to implement the notion of base commitment. In particular, trying to remove negative formulas can be used to realize trying to fulfill base commitments.

Hence, to model a base commitment of deriving a resource or performing an action, we negate formulas of the resource or action. A base commitment to derive resource A is expressed as $A^\bot$. Referring to our example, the base commitment to send two cricket bats to the customer is expressed as $(2\ cri_k\ b@C_c)^\bot$. If the formula of a resource includes time operators, then the corresponding base commitment can be in one of the two possible forms. One is the negation of the whole resource’s formula. A negation of $\bigcirc^nA$ is $(\bigcirc^nA)^\bot$. The second is a formula formed from a negation of the linear logic part, preceded by the same time operators. For example, $\bigcirc^n(A)^\bot$ (or shortly $\bigcirc^nA^\bot$) is the second form. Variations of them can be $\bigcirc^{n_1}(\bigcirc^{n_2}A)^\bot$, where $n_1 + n_2 = n$. Regarding the second form, it can still exert the same effect given a presence of the positive formula of the resource but only at the same time point as of the positive formula.

Negative formulas will not be removed until the corresponding positive formulas are present to remove them or until they expire. However, removal of negative formulas as a result of time passing (unfulfilled commitments) is considered not acceptable for a successful interaction.

In order to fulfill a base commitment, a corresponding positive formula is required such that their multiplicative conjunction yields $\bot$. Indeed, if agent $\alpha$ has a base commitment to bring about resource A or to do action A, represented as $A^\bot$, removing this negative
Commitment based Interaction

formula means deriving the corresponding positive $A$ such that $A \vdash A \top$, and hence fulfilling the base commitment. Similarly to the negative formulas’ cases, the corresponding positive formulas can be in two forms. The first form is a negation of the base commitment’s formula. The second form comprises a negation of the linear logic proposition part of the base commitment’s formula, preceded by the same time operators as of the base commitment’s formula. For example, a base commitment formula $\com{A}$ has two corresponding positive formulas that can remove it. They are $(\com{A})^\bot$ and $\com{A}^\bot$.

Moreover, breakable commitments which are in place to provide agents with the desired flexibility to remove itself from its commitments (i.e. the ability to cancel commitments) can also be modeled naturally in our framework. A base commitment $\com{A}$ to derive $\com{A}$ can be turned into a breakable base commitment as $(\cond{\com{A}})^\bot$. The extra token $\cond{\com{A}}$ and $\bot$ reflect the agent’s internal deliberation about whether the base commitment to derive $\com{A}$ is broken (i.e. if the base commitment is broken then $\cond{\com{A}}$ is derivable and vice versa). Hence, $(\cond{\com{A}})^\bot$ means a base commitment to derive the expected $\com{A}$ or to derive an indication that the base commitment is broken ($\cond{\com{A}}$). After a base commitment $(\cond{\com{A}})^\bot$ is made, in order to fulfill it, the agent can produce $\cond{\com{A}}$ or $\com{A}$. If $\cond{\com{A}}$ is produced, due to the logic deduction $\cond{\com{A}} \vdash (\cond{\com{A}})^\bot$, the commitment $(\cond{\com{A}})^\bot$ is removed without $\com{A}$ being derived. In other words, the base commitment to derive $\com{A}$ is deliberately broken.

In addition, commitments, due to their persistence, make the agents’ actions more predictable. The modeling of base commitments as negative formulas also reflects that. Indeed, under our assumption, all negative formulas are expected to be fulfilled by the respective agents to make the interaction successful. Hence, other agents can reasonably assume that an agent will derive the appropriate resources and actions to fulfil its commitments. As negative formulas remain until the agents fulfill them, they provide a level of predictability.

3.3.3 Goals

In the modeling of goals, we explore the duality and symmetry between what is desired (goals) and what is provided (resources and actions). Hence, goals are also modeled by negative formulas. Such modeling blurs the distinction between goals and base commitments. However, in our scope of research, we only consider those goals that agents are committed to achieving. Therefore, an agent’s goals can also be implemented as commitments to bring about the resources or actions corresponding to the goals. With this approach, goal formulas
can be treated in the same manner as base commitments and become an integral part of the reasoning about resources and commitments.

3.3.4 Conditional Commitments

A conditional commitment [Yolum and Singh, 2002a] requires some conditions to be satisfied prior to making the base commitment(s) active.

Indeed, the relationship between satisfaction of the conditions and the base commitments can be regarded as causal. To model this causal relationship, we make use of the linear implication connective. In particular, \( \rightarrow \) models an automatic and definite transition from a satisfaction of the conditions to the activation of base commitments. A general form is \( \Gamma \rightarrow \Delta \) where \( \Gamma \) is the condition part and \( \Delta \) is the commitment part and typically contains some base commitments. A conditional commitment \( \Gamma \rightarrow \Delta \) of an agent \( \alpha \) is interpreted as that agent \( \alpha \) commits that whenever the condition \( \Gamma \) is satisfied, agent \( \alpha \) will ensure that the commitment \( \Delta \) results. The commitment is enforced by the use of linear implication which makes a definite transition. If the condition is not satisfied, the linear implication can not be applied and hence the commitment \( \Delta \) will not be derived. When its time frame, as specified by its time operators, is exceeded, the conditional commitment becomes invalid.

Commitments to relocate or change ownership of resources can be regarded as a form of conditional commitment whose conditions are that the resources are available at the original places or belong to the original owner. Specifically, if a commitment is to relocate the resource \( A \) from agent \( \alpha \) to agent \( \beta \), we take advantage of using \( \rightarrow \). Given a resource \( A \) at agent \( \alpha \) \( (A@\alpha) \), the commitment \( A@\alpha \rightarrow A@\beta \) will replace \( A@\alpha \) by \( A@\beta \) and consequently relocates resource \( A \) to agent \( \beta \). Similarly, a commitment to change ownership of a resource \( A \) can be modeled as \( A@\alpha \rightarrow A@\beta \). Together, a commitment to both relocate a resource \( A \) and change its ownership is modeled as \( A@\alpha, \alpha \rightarrow A@\beta, \beta \).

We consider conditional commitments in the context of agent interaction. As agents are self-interested, making a promise to another agent of doing some tasks often requires something in return, such as "I promise that if you do something for me, I will do this and that". Hence, conditional commitments from one agent to another often include in the conditions what is required of the other in exchange for what is promised to be done.

For the purpose of supporting modular designs of interactions as series of base and conditional commitments, each conditional commitment (and hence its pre-commitment form) is simply designed to be an independent item in the sense that its application only depends
on the satisfaction of its conditions and does not depend on how other commitments may be satisfied. This design simplifies the negotiation process in that it takes place over one conditional commitment, not a set or sequence of them. This allows us to focus on the mechanism to represent and execute protocols rather than the complexities of negotiation over multiple inter-related items. On the other hand, our mechanism is not restricted to such a simple design but can be extended to cover richer themes of negotiation.

Regarding interaction, a conditional commitment \( \Gamma \rightarrow \Delta \) can be put forward as a proposal from agent \( \alpha \) to agent \( \beta \). The conditional commitment is read as agent \( \alpha \) if agent \( \beta \) makes \( \Gamma \) available for use, agent \( \alpha \) will make the commitment \( \Delta \). When two agents agree on the proposal, the conditional commitment is established between the two agents. In particular, agent \( \beta \) derives \( \Gamma \) and agent \( \alpha \) commits that fulfillment of \( \Gamma \) by \( \beta \) will result in the derivation of \( \Delta \). If \( \Gamma \) is not derived within the valid time frame of the conditional commitment, then the conditional commitment is released.

Some examples of conditional commitments from an agent \( \alpha \) to agent \( \beta \) include:

- If agent \( \beta \) does action \( A \), agent \( \alpha \) will commit to doing action \( B \) (a base commitment \( B^\perp_{\alpha} \) results):

\[
A_\beta \rightarrow B^\perp_{\alpha}
\]

- If agent \( \beta \) brings resource \( A \) to agent \( \alpha \) (from \( A@\beta_\beta \) to \( A@\alpha_\alpha \)), then agent \( \alpha \) will commit to bringing resource \( B \) to agent \( \beta \) at the second next two time point (from \( B@\alpha_\alpha \) to \( B@\beta_\beta \), of which deriving \( B@\alpha_\alpha \) is in a form of a base commitment: \( (B@\alpha_\alpha)^\perp \) )

\[
A@\beta_\beta \rightarrow A@\alpha_\alpha \otimes (\Box^2 B@\beta_\beta \otimes (\Box^2 B@\alpha_\alpha)^\perp)
\]

Moreover, when there is a time bound on the conditional commitment, which is usually described in its time operators, exceeding the time bound will render the conditional commitment invalid. If the agent responsible for fulfilling the condition part fails to do so in time, the conditional commitment is then regarded as being released. Otherwise, the conditional commitment turns into base commitments (a transition from the conditions to the effects takes place) and hence, these base commitments are subject to the time bound instead.

Furthermore, a breakable conditional commitment is modeled as \( A \rightarrow (1 \& B) \) (or \( A \rightarrow (1 \oplus B) \)), instead of \( A \rightarrow B \). When the condition \( A \) is provided, the linear implication brings about \( (1 \& B) \) (or \( 1 \oplus B \)) and it is now up to the owner agent’s internal choice (or external
Commitment based Interaction

choice) whether 1 or B results. If 1 is the outcome, then the conditional commitment is broken.

However, in order to focus on modeling protocols of interaction, we limit our consideration to unbreakable commitments while keeping in mind that it is largely an issue of implementation to include breakable commitments (like the above mentioned modeling) in the modeling framework.

3.3.5 Pre-commitments

We have discussed the modeling of conditional commitments. It still remains to determine how a conditional commitment is formed between two interacting agents. In particular,

- How does an agent decide to make a conditional commitment to another agent?
- How does the agent know that the other agent also wants the conditional commitment and hence is likely to make attempts to fulfill the conditions of the conditional commitment?
- Should the owner stop making a conditional commitment if the other agent does not want the conditional commitment?
- If the owner agent has multiple potential conditional commitments that can serve the same purpose, should it allow other agents to pick the one that they prefer?
- How do we allow an agent to disagree about a conditional commitment and suggest the use of another conditional commitment which might be more suitable to both agents?

Regarding the first question, agents can decide to make a conditional commitment based on whether it can serve a goal or a pending commitment. These goals and commitments can arise from the agents themselves or from other agents in the interaction.

The next questions essentially involve negotiation between agents over some conditional commitments, so that in the end, the chosen conditional commitments are what the owner agent wants to make and also what the other parties want to fulfill their conditions. The involving agents can negotiate over the conditional commitments and alternatives proposed by each other to reach an agreement on one.

During the negotiation phase, conditional commitments are pending (i.e. not yet agreed to). There is a need for another distinct form of conditional commitment to reflect this inactive state due to negotiation. As our approach does not focus on various forms of negotiation
among agents, we make use of the concept of pre-commitment to capture this state, while simplifying the negotiation process.

As discussed in Section 2.3.2, pre-commitment can be thought of as a potential commitment that one agent is willing to commit, but which has not (yet) been accepted by the other agent. This form of pre-commitment is different from the form in which the agent decides in advance that it will commit. Acceptance by the other party is the key to turn the pre-commitment into a conditional commitment.

Both conditional commitments and base commitments can be considered in the form of pre-commitments prior to negotiation. Base commitments are like offers without conditions and hence are assumed to be readily accepted by other agents, especially self-interested agents. Conditional commitments, on the other hand, may require a certain level of contribution on the accepting agents, as reflected in the conditions, and consequently are more likely to involve negotiation to reach agreement among parties. Our work hence focuses on modeling pre-commitments for conditional commitments.

We model pre-commitments in a similar way to commitments but the modeling of pre-commitment also contains universal quantification of variables, which refer to the applicable agents. These agents might be the agents that the conditional commitment is to and/or the agents that the conditions are supposed to be satisfied by. When a pre-commitment becomes a commitment, the variables in the representation of the pre-commitment are assigned to specific agents.

For example, consider a conditional commitment that if a customer pays 10 dollars, the merchant commits to giving the customer a cricket bat. This is expressed as follows

$$10\$@C_c \rightarrow (cricket_b@C_c)^\perp$$

A pre-commitment of the merchant (to any customer) is then expressed using universal quantification over a variable $X$ that represents a customer:

$$\forall X \ 10\$@X_x \rightarrow (cricket_b@X_x)^\perp$$

Note that $X$ is a variable quantified over the domain of agents.

### 3.3.6 Formation of Commitments by Reasoning and Negotiating Pre-commitments

Modeling concepts about agent interaction has been introduced and discussed in previous sections. These concepts include resources, actions, capabilities, base commitments, conditional commitments, goals, and pre-commitments. A summary of these concepts and their
Table 3.1: Summary of Interaction Concepts Modeling

<table>
<thead>
<tr>
<th>Concept</th>
<th>Modeling in TLL</th>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resource</td>
<td>$Time \text{ proposition} @\alpha_\beta$</td>
<td>3.2.1</td>
</tr>
<tr>
<td>Unlimited resource</td>
<td>$!\text{proposition} @\alpha_\beta$</td>
<td>3.2.1</td>
</tr>
<tr>
<td>Status information</td>
<td>$Time \text{ proposition} @\alpha_\beta$</td>
<td>3.2.2</td>
</tr>
<tr>
<td>Action label (no system change)</td>
<td>$Time(\text{preconditions} \rightarrow \text{postconditions})$</td>
<td>3.2.3</td>
</tr>
<tr>
<td>Action (with system changes)</td>
<td>$Time(\text{preconditions} \rightarrow \text{postconditions})$</td>
<td>3.2.3</td>
</tr>
<tr>
<td>Capability</td>
<td>$Time(\text{preconditions} \rightarrow \text{postconditions})$</td>
<td>3.2.3</td>
</tr>
<tr>
<td>Base commitment</td>
<td>$\text{formula}^-$</td>
<td>3.3.2</td>
</tr>
<tr>
<td>Goal</td>
<td>$\text{formula}^-$</td>
<td>3.3.3</td>
</tr>
<tr>
<td>Conditional commitment</td>
<td>$Time(\text{preconditions} \rightarrow \text{postconditions})$</td>
<td>3.3.4</td>
</tr>
<tr>
<td>Pre-commitment</td>
<td>$\forall X[Time(\text{preconditions} \rightarrow \text{postconditions})]$</td>
<td>3.3.5</td>
</tr>
</tbody>
</table>

Legend: $Time$: a combination of temporal operators $\bigcirc$, $\square$, and $\diamond$; $\text{proposition}$: TLL proposition; $\text{formula}$: TLL formula; $\alpha, \beta$: names or IDs of agents; $\text{preconditions}$, $\text{postconditions}$: multiplicative conjunctions of TLL formulas.

modeling in TLL is provided in the table 3.1. More precise notations of the modeling are provided in a grammar formulation form in Section 4.2.1. It can be seen that the notations of various concepts are similar, which reflects the pragmatic similarities of these concepts with respect to resource handling. For example, as also mentioned in their modeling discussion, actions that affect resources and system properties have similar effects to capabilities. Reading status information can be thought as using a kind of resource. Moreover, our modeling focuses on the dynamic interactions and relationships between them so that capabilities and actions are considered with respect to the effects they have on resources and state properties; base commitments are modeled as dual to the resources and actions required to fulfill them; and pre-commitments and conditional commitments emerge from capabilities. Overloading notations, however, has its own issues as discussed in Section 8.2.3 and can be can be overcome by instrumenting extra (uncomplicated) steps in handling the notations.

We then describe further interactions among these concepts in the presence of pre-commitments. In particular, we examine how agents utilize their sets of pre-commitments.

In general, pre-commitments can be regarded as services that agents can provide to other agents. In these services, the owner agent performs transformations from pre-conditions to post-conditions. To use a service, an agent has to provide the pre-conditions. Pre-commitments can then be used in a proactive manner in which the agents propose them to
other agents, given that these pre-commitments upon being resolved can lead to fulfillment of their goals and commitments. Pre-commitments can also be used in reactive manner in which the agents propose them as a (helpful) response to requests from other agents.

Either way, agents need to be able to determine and act on the relevance of pre-commitments with respect to a goal or a commitment or a request. This relevance can be modeled simply as matching what the pre-commitment can provide with what the goal/commitment demands. Such matching is described in detail in Section 4.3.7. A proposed pre-commitment that is determined to be irrelevant will be rejected and hence no corresponding commitment is formed. Another pre-commitment may then be proposed. If a pre-commitment is found relevant by the receiving party, it is accepted and this acceptance allows the owner of the pre-commitment to make a conditional commitment out of the pre-commitment. Only then, a conditional commitment is formed and binds the two agents. The receiving agent is supposed to satisfy the conditions of the conditional commitment. For the owner (proposing) agent, it is tied to the commitment that upon the conditions being satisfied, the owner agent fulfills the commitment part of the conditional commitment.

3.3.7 Evolution of Commitments

It can be seen that conditional commitments can be based on the agents’ capabilities. Capabilities of agents can be viewed as services that the agents provide to other agents. As agents are self-interested, for each capability that an agent can perform for others, it can attach a price that the other has to pay in exchange. The price can be given in terms of money or resources. To ensure that the capability will be performed when its price is paid, the concept of commitment can be used. In particular, from a capability that an agent can perform as a service, information about the price is attached as conditions to form a conditional commitment that the agent can make to other agents. To further support negotiation over conditional commitments, a pre-commitment is used and formed out of the capability. Pre-commitments show a willingness to commit to performing the capabilities under the specified conditions. This translation from a capability to a pre-commitment makes the agent able to provide the capability as a service during interaction.

The evolution of commitments is described as follows. Agents have capabilities, goals and possibly some commitments to be fulfilled. Those capabilities that can used to serve others are put in a form of pre-commitments. As agents interact, they can be proactive by proposing their pre-commitments to other agents if these commitments can help to fulfill their goals or
commitments. Also, they can react to requests by proposing pre-commitments that can help to satisfy the requests. Conditional commitments emerge from the agreed pre-commitments. The conditions and possibly base commitments of conditional commitments need their corresponding positive formulas (i.e. desired resources and/or actions) to be derived. Once these positive formulas are provided, they will resolve the conditional commitments, possibly into base commitments and subsequently remove all the base commitments.

3.3.8 Conventions

The modeling of breakable base commitments and conditional commitments has been discussed in Sections 3.3.2 and 3.3.4. However, to focus on demonstrating the modeling of flexible interactions based on the notion of commitment, we limit the scope of our work to deal with those situations where base commitments and conditional commitments cannot be broken. Handling of broken commitments with penalties and compensation is one item of further work.

3.3.9 Rules for Protocols

Protocols are commonly specified in terms of rules for interaction. In particular, each rule guides the participating agents on their interactive actions. Agents then interact by utilizing the pre-determined rules to fulfill their goals and objectives. In our approach, pre-commitments and capabilities can also be viewed as rules for interaction.

We define a rule as an encoding of the form \(\text{time} \_\text{op} \ [\Gamma \rightarrow \Delta]\), where \(\text{time} \_\text{op}\) refers to a temporal operator or a combination of operators and \(\Gamma, \Delta\) refer to TLL formulas. With this encoding, rules can be regarded as capabilities, conditional commitments and even some actions.

Each rule can also be regarded as a specification of a transformation process which describes the conditions before (\(\Gamma\)) and after (\(\Delta\)) as a result of executing the rule. When a rule is applied, it takes an appropriate condition \(\text{time} \_\text{op} \ \Gamma\), and replaces it with the post-condition \(\text{time} \_\text{op} \ \Delta\).

Rules can be classified as local or interaction, according to its scope of use. Local rules are not used directly for interacting with other agents (such as answering a request) but can be used internally by the owner agent. Interaction rules are used to guide the interaction with other agents. In interaction rules, there are variables quantified over the domain of agents. An assignment of these variables to agents determines the participating agents in
the interaction with the owner agent. The other agents are expected to play some roles in the rules. In our case, they are expected to fulfill the conditions of the rules and the owner agents are expected to deliver the outcomes by applying the rules.

It can be seen that local rules correspond to capabilities, which can be used internally. Interaction rules correspond to pre-commitments. However, if variables in pre-commitments are assigned to the owner agent, these pre-commitments are then local rules as the rules are carried out entirely by the owner agent.

An example of interaction rules of an agent α is

$$\Box[\forall X \ 20\$@X \rightarrow ticket@X \otimes ticket@\alpha^\perp \otimes 20\$@\alpha]$$

in which X is an agent variable. The rule allows agent α to sell its ticket to any agent who is willing to pay 20 dollars. By assigning $X = \beta$ during an interaction with agent β, the rule becomes

$$\Box[20\$@\beta \rightarrow ticket@\beta \otimes ticket@\alpha^\perp \otimes 20\$@\alpha]$$

in which agent β is responsible to pay 20 dollars and as a result of such payment, agent α will commit to providing the ticket and agent β will receive the ticket.

Applications of interaction rules can be dynamic as rules can be combined or extended without changing provability. Some examples of rule manipulation are:

- \((A \rightarrow B) \otimes (C \rightarrow D) \vdash (A \otimes C \rightarrow B \otimes D)\)
- \((A \rightarrow B) \& (A \rightarrow C) \vdash A \rightarrow (B \& C)\)
- \(A \rightarrow B \vdash A \& C \rightarrow B\)
- \(A \rightarrow B \vdash A \rightarrow (B \oplus C)\)
- \((A \rightarrow C) \& (B \rightarrow C) \vdash (A \oplus B) \rightarrow C\)

### 3.3.10 State

As we are not concerned with BDI agents per se, but with the planning and utilization aspect of resources and actions, we are more concerned with which resources are consumed and produced, and which actions are carried out during the agent’s lifetime. We limit the concept of a state of an agent to capturing the status of these. Similar work in this direction, [Küngas, 2003] and [Küngas, 2004b] also consider states of agents in terms of resources.
The state of an agent in our framework includes all resources that the agent has, actions and capabilities that can be performed by the agent and the agent’s commitments. As the multiplicative conjunction $\otimes$ allows specification of items without any constraints on their order, we can represent a state by a multiplicative conjunction of TLL formulas representing resources, actions, capabilities and commitments. For example, a merchant’s state might be having 50 dollars and 100 junior cricket bats which is represented as $50\$@M \otimes 100jr\_cb@M$. Moreover, being connected via $\otimes$, items of states can also be used in independent contexts, which supports concurrent utilization.

State changes of agents can take place as a result of actions, applications of capabilities, resolution of commitments or time passage. For instance, a successful application of a capability of the form $\Gamma \rightarrow \Delta$ subtracts formulas in $\Gamma$ and the capability’s formula from the set of state formulas and adds to it the formula $\Delta$. Moreover, when the time moves to the next time point, all the formulas associated only with the present (excluding those with $\Box$ and $\Diamond$ operators), are removed.

Moreover, because agents are operating in an environment, they constantly receive inputs from the environment. Such inputs, for example, are obtained either by agents actively collecting information through their observations or getting feedback from the environment. The inputs that are relevant to our context of modeling interaction can be classified in terms of status information, resources, commitments and effects of actions or capabilities being performed. Given our focus on modeling interaction, we do not provide a specific agent mechanism for receiving and processing environment inputs but assume a typical environment input processing mechanism for agent systems to be in place. Such a mechanism also updates agents’ states and hence affects agent interaction processes. In order to make sure that the updates are reflected in agent interaction, in our specification and execution framework for interaction, the states of agents are always referred to as the most current and up-to-date states. Given that state formulas are connected via $\otimes$ connective, adding new formulas to the states is simple. Removing obsolete formulas can also be done easily using the $\rightarrow$ connective via applications of actions, capabilities, resolution of commitments, or time passage. Hence, the mechanism to keep the agents’ states updated remains largely an implementation issue and hence is not discussed in detail in the thesis.
3.3.11 Cricket Bat Sale Example

We provide an illustration of what a pre-commitment is and how it evolves into a conditional commitment and then is fulfilled with respect to our example of cricket bat sale (section 3.1).

In the example, the merchant pre-commits that given that the customer pays 20 dollars via Paypal or by credit card and provides an address, the merchant will give the customer 2 junior cricket bats and a gift. Note that as a result of fulfilling the pre-commitment, the merchant receives 20 dollars and the customer’s address, and have the internal commitment of getting 2 junior cricket bats and a gift ready.

\[
20 \Box [(20\$@C_C \otimes (\text{via}_{\text{PP}}@M_C \oplus by_{\text{cred}}@M_C) \otimes \text{addr}@C_C)]
\]

\[
\rightarrow (20 \Box @M_M \otimes \Box 2jrcb@C_C \otimes 2jrcb@M_M \otimes \Box gift@C_C \otimes gift@M_M \otimes \Box \text{addr}@M_C])
\]

A pre-commitment, like that of the merchant, can be proposed to the involving party and once it gets accepted, it becomes a conditional commitment by the owner agent. Once the conditions are provided, for example, \((20\$@C_C \otimes (\text{via}_{\text{PP}}@M_C \oplus by_{\text{cred}}@M_C) \otimes \text{addr}@C_C)\) are provided, the owner agent is expected to fulfill the conditional commitment, which means that

\[
(20 \Box @M_M \otimes \Box 2jrcb@C_C \otimes 2jrcb@M_M \otimes \Box gift@C_C \otimes gift@M_M \otimes \Box \text{addr}@M_C])
\]

are derived. The outcome might contain internal commitments, like \(2jrcb@M_M \otimes gift@M_M\), which further require to be fulfilled by the owner agent (in this case, the merchant).

3.4 Summary

Chapter 3 has described how various agent interaction concepts such as resources, resource location, resource ownership, resource exchange, action, capability, goal, base commitments, conditional commitments and pre-commitments are modeled. We also mentioned how modeling of such concepts might be put into work in the context of resource-based and commitment-based interaction. In the next chapter, we will discuss a specification framework of agent interaction based on such modeling.
Chapter 4

Specifying Flexible Interaction

In the context of open systems, agents may have different designs and architectures. To support such diversity, it is important to make the specifications of agent interactions independent of the agents’ internal designs and architectures. Hence, the only aspect of an agents’ internal systems that concerns is the parts that can be utilized for agent interaction. These include resources, actions, and capabilities of agents.

We take the approach that interactions are driven by the agents’ goals and commitments. However, we do not impose the notion of shared goals or joint goals among agents. This facilitates the notion that interaction is at the agents’ discretion and is not imposed on agents as a means to achieve shared or joint goals.

As discussed in Chapter 3, interaction between agents can be structured in terms of commitments. Commitments can occur in various forms such as pre-commitments, conditional commitments, internal commitments, and base commitments. Pre-commitments are specified as services that an agent can be offer to other agents. Conditional commitments result from the pre-commitments that are agreed upon by the involving parties. Internal commitments may arise as a result of interaction and can be regarded as a form of goals. Base commitments can result from resolving conditional commitments or be part of the specification of interaction.

In our framework, given the assumption that goals are also treated as internal commitments, commitments in general are what govern agent behaviors. Commitments influence agents’ decisions in selecting actions, transformations and exchanges of resources as well as further commitments. Moreover, resolving commitments and fulfilling requests from other agents may result in some further commitments being made. This process goes on until all
goals and commitments are fulfilled.

4.1 Specification of Interaction

A protocol can be defined in terms of a set of active commitments among participating agents. In our approach, we make use of the notion of pre-commitment and a protocol specification includes specifications of pre-commitments at each participating agent. Moreover, as interaction is driven by the agents’ goals and commitments, we consider specification of an interaction in a broader context. In particular, a specification of interaction includes specifications of the protocols and specifications of the agents’ goals and base commitments. While goals can be regarded as internal to agents, base commitments are a public responsibility that agents are expected to fulfill. Interaction is then guided by an agent’s goals (internal commitments) and base commitments, and the logic of the protocol (or rules of encounter) is embodied in the specified pre-commitments.

Note that the specification of protocols (as a set of pre-commitments) in our approach is an inseparable part of the specification of agent interaction. From now on, for simplicity of discussion, unless otherwise stated, we will use the broader term specification of interaction, on the understanding that this includes protocol specification.

Specification at each participating agent reflects their respective roles in the intended interactions. The role of each agent is detailed via the goals that the agent aims to achieve, the base commitments the agent is required to fulfill, and the services (pre-commitments) the agent can provide. For example, in our cricket bat sale example, the role of the customer manifests itself via its goal of obtaining cricket bats and a gift, and that it can provide money and information to facilitate payment for the goods it purchases.

The specification at each participating agent is considered as private information, accessible only to that agent. Agents then discover other agents’ goals, commitments and services via interaction. This design allows agents to avoid disclosing private information to the advantage of others. Indeed, maintaining such privacy is desirable among self-interested agents and promotes agent autonomy over the control of information regarding its goals, internal commitments and pre-commitments.

Our specification allows goals and base commitments to be fulfilled internally by the agents, i.e. without interaction. This increases the flexibility of interaction, as commitments and goals can either be fulfilled internally or via interaction. How a commitment or goal is fulfilled will depend on a number of factors, such as the internal resources and actions that
are available, the preferences of agents about interaction, what others are willing to offer as services, and changes due to the environment or other interactions.

Our specification can also be viewed as a rule-based approach in which interaction protocols are specified by defining interaction rules for each participating agent. Each agent is described in terms of its resources, commitments (goals) and capabilities. Interaction rules are structured as “conditions → effects”, so that successful applications of the rules will replace the condition formulas by the effect formulas and hence drive state changes. Interaction rules provide constraints on how the respective agents might interact with other agents and are specified to naturally reflect the agents’ capabilities specific to their role in the interactions. They are specified in terms of which resource transformations, resource exchanges or actions are to be carried out, and how existing commitments can be fulfilled or new commitments emerge.

Furthermore, it can be seen that rather than specifying protocols as a set of commitments, our approach specifies protocols as a set of pre-commitments and allows commitments to be dynamically formed when the agents utilize the pre-commitments to fulfill their goals and commitments. Our approach hence makes clear the separation between the agents’ goals and commitments and the ways the agents interact with each other (as specified in pre-commitments) to achieve them. Hence, it can be seen that interaction emerges from a set of suitably chosen commitments rather than from a pre-determined set. In other words, protocols are dynamically and flexibly utilized.

4.2 Designing Interaction Specification

4.2.1 The Specification Language

The formulas that represent resources, actions, capabilities, base commitments and conditional commitments are defined as follows.

Each resource can be represented by a proposition or an atomic formula preceded with temporal operators. Let $A$ be an atom in TLL. Let $@α$ and $−y$ be markers that represent agent location and ownership of the resource formulas respectively, as described in 3.2.1. Let $x, y$ be variables in the domain of agents. For example, a value of $x$ can be agent $α$. The notation $A@x_\alpha$ means that formula $A$ is at agent $x$ and is owned by agent $y$.

Relationships between resources are co-existence or mutual exclusiveness due to internal choices or indeterminate possibilities. Having no resource is expressed by the constant 1. Resource formulas together with their relationships then have the following grammar:
Each action is represented declaratively and hence in a similar way as resources (except for the constant 1). Action formulas have the following grammar.

\[
AC ::= A@x \| AC| \square AC| \Diamond AC| AC \otimes AC| AC & AC| AC \oplus AC
\]

Capabilities are modeled with respect to their effects and hence can be a resource transformation or state update. The transformation and updating processes are modeled by the connective $\rightarrow$ and the formulas are possibly prefixed by $\bigcirc$ and/or $\square$. Because it does not seem natural to have a capability over which the agent has no control and no information about when it can be used, we do not use $\Diamond$ to prefix a capability formula. Hence, capability formulas have the following grammar:

\[
CA ::= R \rightarrow R| \bigcirc CA| \square CA
\]

Base commitments are modeled as a negative form of resources and actions formulas and have the grammar:

\[
B ::= R \rightarrow \bot
\]

Conditional commitments are modeled as a linear implication from pre-conditions to post-conditions (the commitment part). Such pre- and post- conditions can be specified in terms of resources, actions and base commitments which can be combined together via the connective $\otimes$. Hence, conditional commitments formulas have the following grammar:

\[
CC ::= C \rightarrow C| \bigcirc CC| \square CC
\]

where C is a pre- or post-condition formula, as in the grammar below.

\[
C ::= B|R|C \otimes C
\]

Pre-commitments are defined as potential conditional commitments. Hence, their formulas can be conditional commitments with some variables over the domain of agents. Pre-commitments formulas then have the following grammar:

\[
P ::= CCx
\]
where $CCx$ is similar to $CC$ but contains at least one agent variable.

State formulas can be formulas of resources, actions, capabilities, base commitments or a combination (multiplicative conjunction) of states. Hence, state formulas have the grammar:

$$S ::= \bot | R|AC|CA|B|S \otimes S$$

We also define basic TLL formulas as formulas of the form:

$$\bigcirc^n A, 0 \leq n$$

where A is an atomic LL formula.

We denote $R$(rule) as the formulas on the right hand side of the linear implication and $L$(rule) as the formula on the left hand side.

Examples of various forms of formulas are as follows.

The merchant Mer has 200 junior cricket bats and 300 heavy cricket bats available any-time. They are formulated as $200 \Box jr\_cb@M_m \otimes 300 \Box heavy\_cb@M_m$

The merchant Mer can carry out the action of issuing quote of sale: $issue\_quote@M_m$

The merchant Mer also has a capability of informing the customer Cus if the Cus’s credit check result is a failure:

$$\eta \Box [cred\_disappr@M_m \rightarrow cred\_disappr@C_c]$$

A pre-commitment of the merchant Mer that can be offered as a sale is

$$\Box[\forall X, 20\$@X \otimes (via\_PP@M_x \oplus by\_cred@M_x) \otimes addr@X]$$

$$\rightarrow 20 \Box \$@M_m \otimes 2jr\_cb@X \otimes 2jr\_cb@M_f \otimes \Box gift@X \otimes gift@M_f \otimes \Box addr@M_x$$

When this pre-commitment is accepted as a deal between the merchant Mer and the customer Cus, the corresponding conditional commitment is as follows:

$$\Box[20\$@C_c \otimes (via\_PP@M_c \oplus by\_cred@M_c) \otimes addr@C_c]$$

$$\rightarrow 20 \Box \$@M_m \otimes 2jr\_cb@C_c \otimes 2jr\_cb@M_f \otimes \Box gift@C_c \otimes gift@M_f \otimes \Box addr@M_c]$$
4.2.2 Designing Goals

As discussed above, we do not use the notion of shared goal or joint goals nor goals of interaction but consider individual goals of agents. As we focus on interaction, we limit our consideration to those individual goals which interaction between agents can help to achieve. In Section 3.3.3, we discuss a simplification of the modeling of agents’ individual goals by treating goals as a form of internal commitments. Goals are then formed as a dual to what is provided to fulfill the goals and hence are negative formulas.

4.2.3 Designing Pre-commitments

A pre-commitment is comprised of pre-conditions and post-conditions. On the one hand, a pre-commitment can act as a form of capability that the owner agent can utilize together with its other resources, actions and other capabilities. On the other hand, pre-commitments can be designed as a constructing unit of agent interaction.

In particular, a pre-commitment can encode an exchange of goods or a sale. The pre-conditions then include the resources and actions required for the exchange or the price that agent has to pay to obtain the goods. The post-conditions are the resources and actions or the goods to be purchased that the owner agent are required to provide.

Moreover, a pre-commitment can encode a resource transformation process in which two agents can participate. The pre-conditions include the resources and actions required of one partner and/or the price its has to pay to use the process. The post-conditions include the resources and actions required of the other partner (the owner agent) as well as the outcomes of the process. The outcomes may also include gains for many agents.

Furthermore, a pre-commitment can be utilized as a service provided by one agent to another agent. In this service, the agent provides certain outcomes and commitments as described in the post-conditions at the cost of what is described in the pre-conditions. As discussed in Section 3.3.4, our approach aims to design conditional commitments in a modular fashion. Hence we also aim for a modular design for pre-commitments. Specifically, this means that what is required in the pre-conditions is solely an exchange for the transformation service plus any outcomes and commitments of the owner agent in the post-conditions. Indeed, such an assumption also facilitates independent use of pre-commitments by their owners in interaction and hence enables a modular approach to utilization of pre-commitments.

As agents can be viewed as possessing resources and capabilities, it is natural to design pre-commitments based on an agent’s capabilities. We discuss a design process that translates
a capability to a pre-commitment.

Firstly, a capability $\Gamma \rightarrow \Delta$ of agent $\alpha$ is turned into a deal that the owner agent can offer to another agent. The pre-conditions part then specifies requirements of resources and actions as well as monetary units that an agent has to fulfill for one part of the deal. Monetary units refer to the price an agent has to pay for obtaining the outcomes or goods in the post-conditions and/or using the service. The post-conditions part then specifies the outcomes of the exchange or transformation for each agent, including any further base commitments of the owner, which together form the other part of the deal.

Given a capability $\Gamma \rightarrow \Delta$ of agent $\alpha$ which can be used as a service to some agent $X$, we denote what is required of $X$ as $\Gamma_1$ and what is provided by $\alpha$ as $\Gamma_2$. Apart from $\Gamma_1$, what $X$ has to pay for the service is denoted as $P$. Hence, $\Gamma$ is comprised of $\Gamma_1$, $\Gamma_2$ and $P$. Moreover, as a result of paying $P$ and deriving $\Gamma_1$, $X$ receives $\Delta_1$, which is part of what is produced ($\Delta$) by the capability. We denote $\Delta_2$ as what the owner agent $\alpha$ gets initially from the capability. $\Delta$ is then comprised of $\Delta_1$, $\Delta_2$ and $P$. Attaching further information about location and ownership, the capability then becomes

$$P@X_X \otimes \Gamma_1@X_X \otimes \Gamma_2@\alpha_\alpha \rightarrow \Delta_1@X_X \otimes \Delta_2@\alpha_\alpha \otimes P@\alpha_\alpha$$

where $P@X_X$ denotes the price $X$ pays and $P@\alpha_\alpha$ denotes what $\alpha$ receives as a result.

To turn the capability into a pre-commitment of the agent $\alpha$ to be offered to the agent $X$, which is also of the form $\Gamma' \rightarrow \Delta'$, we match the pre-conditions $\Gamma'$ with the price $P@X_X$ and the resources and actions $\Gamma_1@X_X$ to be provided by the agent $X$ and turn $\Gamma_1@\alpha_\alpha$ into an internal commitment required at the agent $\alpha$:

$$P@X_X \otimes \Gamma_1@X_X \rightarrow (\Gamma_2@\alpha_\alpha)^\bot \otimes \Delta_1@X_X \otimes \Delta_2@\alpha_\alpha \otimes P@\alpha_\alpha$$

Note that this translation still preserves what is demanded in the capability, which now contains $P@X_X \otimes \Gamma_1@X_X$ and an internal commitment $((\Gamma_1@\alpha_\alpha)^\bot)$ of deriving $\Gamma_1@\alpha_\alpha$ for the agent $\alpha$ to fulfill.

Moreover, the translation to a pre-commitment does not prevent the capability from being used internally. Indeed, the owner agent can turn a pre-commitment to a conditional commitment to itself. In doing so, as in the above example, the above agent $X$ can be regarded as a variable and can be assigned as $\alpha$. The conditional commitment becomes:

$$P@\alpha_\alpha \otimes \Gamma_1@\alpha_\alpha \rightarrow (\Gamma_2@\alpha_\alpha)^\bot \otimes \Delta_1@\alpha_\alpha \otimes \Delta_2@\alpha_\alpha \otimes P@\alpha_\alpha$$
\( P@\alpha_n \) is still required but will be returned afterward, meaning that there is no extra cost. Hence, the conditional commitment can be thought of as
\[
\Gamma_1@\alpha_n \rightarrow (\Gamma_2@\alpha_n)^+ \otimes \Delta_1@X \otimes \Delta_2@\alpha
\]
while the agent \( \alpha \) makes a debt of \( P@\alpha_n \) and immediately returns the debt after fulfilling the commitment.

Note that regarding those base commitments in the post-conditions that involve changing resource location and/or ownership, we can simply specify them as internal commitments to derive the respective resources. The tasks of changing resource location and/or ownership can then be specified as having the resources with the expected location and/or ownership in the post-conditions. Such modifications still produce the same effects when the conditional commitment is resolved.

Furthermore, this translation from capabilities to pre-commitments conveys an important property. Unlike capabilities, pre-commitments have pre-conditions which contain only what is required of the recipient agent. What is required of the proposing agent in pre-commitments is not put in the pre-conditions but in the post-conditions. Hence, there is no guarantee in pre-commitments that what is required of the proposing agent will be fulfilled. This lack of certainty directly reflects the nature of conditional commitments as promises but not their actual fulfillment.

In addition, while capabilities can be regarded as what an agent can perform, pre-commitments can be considered as capabilities that the agents can perform together plus information on the distribution of tasks, costs (prices to pay) and outcomes among participating agents. As can be seen from the translation procedure from a capability to a pre-commitment above, these steps can be automated. To derive a fair distribution of tasks, costs and outcomes, agents should be equipped with information on their utility values and strategies on dealing with them. This is beyond the scope of the thesis and hence we assume the tasks of designing pre-commitments is done by humans.

### 4.2.4 Specification for Cricket Bat Sale Example

We will illustrate the design of specifications of agent interaction via an example below. The interaction scenarios of concern are described in Section 3.1. We then describe how the specification of the interaction can be done based on the agents’ capabilities which are described in Section 3.2.5. The specification of the cricket bat sale example includes the
resources, capabilities, actions, goals and pre-commitments of the participating agents. The specification of resources, actions, and capabilities was shown in Section 3.2.5. In this section, we will show the specification of goals and pre-commitments which are derived from the corresponding capabilities. Pre-commitments of the agents in the example are designed from capabilities 1, 2, and 5. Other capabilities are used internally by the owner agents.

**At agent Mer**

In the example, some capabilities of participating agents can be turned into pre-commitments by the owner agent.

Given capability 1

\[
\eta \Box [10 \mathcal{R}_C \otimes (\text{via}\_PP\mathcal{R}_M \oplus \text{by}\_cred\mathcal{R}_M) \otimes \text{addr}\mathcal{R}_C \otimes jr\_cb\mathcal{R}_M] \\
\rightarrow 10 \Box [\mathcal{R}_M \otimes \Box jr\_cb\mathcal{R}_C \otimes \Box \text{addr}\mathcal{R}_C]
\]

the merchant pre-commits that if the customer \(X\) pays 10 dollars via Paypal or by credit card and provides its address, the customer will receive a junior cricket bat. The pre-conditions are the responsibility of the customer \(X\) in which \(X\) needs to reserve 10 dollars of its own and at its place \((10 \mathcal{R}_X)\) to pay via a payment method to Mer \((\text{via}\_PP\mathcal{R}_X \text{ or by}\_cred\mathcal{R}_X)\), and to send its own address \((\text{addr}\mathcal{R}_X)\) to Mer. The post-conditions then contain the resulting resources for the customer \((\Box jr\_cb\mathcal{R}_X)\) and for the merchant \((\Box 10 \mathcal{R}_M, \text{ and } \Box \text{addr}\mathcal{R}_C)\). The post-conditions also include the commitment of the merchant as part of the deal \(jr\_cb\mathcal{R}_M\). The pre-commitment is expressed as interaction rule 1 below

**Rule 1:**

\[
\eta \Box [\forall X, 10 \mathcal{R}_X \otimes (\text{via}\_PP\mathcal{R}_X \oplus \text{by}\_cred\mathcal{R}_X) \otimes \text{addr}\mathcal{R}_X] \\
\rightarrow 10 \Box [\mathcal{R}_M \otimes \Box jr\_cb\mathcal{R}_X \otimes jr\_cb\mathcal{R}_M \otimes \Box \text{addr}\mathcal{R}_X]
\]

Similarly, capability 2

\[
20 \Box [(20 \mathcal{R}_C \otimes (\text{via}\_PP\mathcal{R}_M \oplus \text{by}\_cred\mathcal{R}_M) \otimes \text{addr}\mathcal{R}_C) \otimes 2jr\_cb\mathcal{R}_M \otimes gift\mathcal{R}_M] \\
\rightarrow (20 \Box [\mathcal{R}_M \otimes \Box 2jr\_cb\mathcal{R}_C \otimes \Box \text{gift}\mathcal{R}_C \otimes \Box \text{addr}\mathcal{R}_C])
\]
is turned into the following pre-commitment in which Mer pre-commits that if the customer pays 20 dollars via Paypal or by credit card and provides its address, Mer will give the customer 2 junior cricket bats and a gift.

**Rule 2:**
\[
\text{20} \big[ \forall X, (20\$@X_X \otimes (\text{via}_P M_X \oplus by_c@M_X) \otimes addr@X_X) \\
\rightarrow (20 \$@M_M \otimes \Box 2\text{jr}_cb@X_X \otimes 2\text{jr}_cb@M_M \otimes \Box gift@X_X \otimes gift@M_M \otimes \Box addr@M_X)]
\]

**At agent Ebank**

**Rule 5:** from capability 5, a pre-commitment is designed in which EBank pre-commits that if a quotation and credit number of the customer Y are given by the merchant X, it will either arrange credit payment or inform a disapproval of credit payment.

\[
\eta \big[ \forall X, \forall Y, \text{quote}@B_X \otimes \text{cred}_no@B_Y \otimes \text{quote}_req@X_B \otimes \text{cred}_paym@X_B \otimes \Box \text{cred}_disappr@X_B]
\]

**At agent Cus**

Cus has a goal of obtaining 2 junior cricket bats and a gift at some time. They are supposed to be obtained at the same time:

\[
[\diamond (2\text{jr}_cb@C_C \otimes gift@C_C)]^\perp
\]

The specifications of resources, capabilities and actions in Section 3.2.5 and the specifications of pre-commitments (which replace the corresponding capabilities) and goals above together form a specification of interaction of the cricket bat sale example. Given such a specification of agent interaction, the next question is how to turn the specification into interaction, which is discussed in the next section.

### 4.3 From Specification to Execution

Agents interact with each other by sending and receiving messages in the form of a request, a proposal, an acceptance, , a rejection or a failure notice. Request messages typically contain goals or base commitments while proposal messages are pre-commitments. The way in which
agents make a request or a proposal or respond to these messages are guided by an interaction model discussed below.

4.3.1 Interaction Model

Overall, agents carry out interaction specifications by exchanging messages according to some interaction model. To serve the purpose of demonstrating how interaction is constructed based on specifications in our framework, we keep the interaction model simple. It has the following properties.

When an agent can not achieve a commitment or a goal by itself, it will make a request for that commitment or goal to an appropriate agent. The requested agent searches for a relevant pre-commitment of its own to propose to the requesting agent. If the search can not find any, a failure notice will be returned to the requesting agent.

When a proposal is received, the recipient checks if the respective pre-commitment is relevant to any of its current goals or commitments. If so and the recipient cannot achieve these goals or commitments by itself, then it will send a message of acceptance to the proposing agent. Otherwise, the recipient will send a message of rejection.

Once the proposal is accepted, a conditional commitment is formed by the proposing agent to the recipient agent out of the pre-commitment of the proposal. When the recipient fulfills the conditions in the proposal, the proposing agent will fulfill its respective commitments. If the recipient agent does not satisfy the requirements, then the conditional commitment remains inactive. Fulfilling the conditions and commitments of the proposal may in turn lead to further interaction.

We discuss in detail the format of messages and issues related to request and proposal messages, as well as responses to a request and a proposal and relate these processes to the process of fulfilling commitments.

4.3.2 Message Format

To indicate the source and destination of messages, we use “Source to Destination:” prior to each message, where Source refers to the sender agent and Destination refers to the recipient agent. For example “Cus to Mer:” denotes that the message is sent from the agent Cus to the agent Mer.

Request messages start with the key word REQUEST. The format is “REQUEST formula”, where formula in request messages normally refer to base commitments and hence is a neg-
ative formula. An example is a request message of a junior cricket bat from a customer to the merchant:

\[ \text{Cus to Mer: REQUEST } jr \_ cb \] 

Similar to request messages, proposal messages commence with “PROPOSE”. Formulas of proposals are typically pre-commitments. The form of proposal is:

\[ \text{PROPOSE } \bigcirc^n(\Gamma \rightarrow \Delta) \text{ or PROPOSE } \Box(\Gamma \rightarrow \Delta) \]

The time frame of the proposal is defined as a particular time point \( \bigcirc^n \) for the proposed formula \( \bigcirc^n(\Gamma \rightarrow \Delta) \) and as a range of time points, starting from now if the proposed formula is \( \ Box(\Gamma \rightarrow \Delta) \). The time when the implication of the proposal’s formula is carried out is within this time frame.

In order to reply to a request, an agent sends back a proposal or a failure notice. A failure notice has the form:

\[ \text{REQUEST } formula \text{ FAILS} \]

where \( \text{REQUEST } formula \) is the original request.

To make a response to a proposal, an agent indicates an acceptance by sending

\[ \text{PROPOSE } formula \text{ ACCEPT} \]

or sends a rejection

\[ \text{PROPOSE } formula \text{ REJECT} \]

where "\( \text{PROPOSE } formula \)" is the original proposal.

After an agent accepts a proposal, if the agent does not fulfill the proposal’s requirements, it sends back a message to indicate this.

\[ \text{PROPOSAL } formula \text{ FAILS} \]

where \( formula \) is the formula of the original proposal.

Note that our approach does not specify the full details of message format, which is a typical part of designing an ACL and hence is beyond the scope of research.
4.3.3 Request Messages

A request message is formed out of a base commitment of an agent for several reasons.

The agent cannot find a way to fulfill the base commitment internally using its own resources and internal capabilities and hence needs to send a request to seek outside possibilities.

Alternatively, the agent may prefer fulfilling the base commitment via interaction with other agents to explore others’ resources and capabilities and hence sends out the request.

Even if some of its pre-commitments can be used to fulfill partly or wholly the base commitment, rather than using it, the agent can still send a request for the base commitment to other agents if it prefers to do so.

Indeed, the decision to send a request is an internal deliberation process of the agent and is not within our scope. We assume that agents can decide by themselves if they should form a request from an unfulfilled base commitment.

The request formula may contain multiple base commitments (negative formulas) connected via multiplicative conjunction (⊗) or additive connectives (&, ⊕). For simplicity, however, we assume that each request is associated with only one base commitment. This does not place a limit on what agents can request as they can deploy a mechanism to break up a combination of base commitments into individual ones. Such a mechanism can be based on the work on splitting up formulas in Section 6.2. Under this assumption, each request contains only a single negative formula.

Once forming a request message, the owner agent considers which agent to send it to. However, knowing which agent is relevant requires some knowledge of other agents’ resources and capabilities. We assume that agents do not know this information and hence agents send request messages to all other agents in broadcast mode to explore all the possibilities. This model also allows the sender agent to explore in parallel multiple opportunities with different agents, which is useful when the whole request cannot be fulfilled by a single agent.

To respond to a request from another agent, an agent does not try to fulfill the commitment in the request using its own private resources, actions and capabilities but looks for a deal to offer from its pre-commitments. If a pre-commitment is found relevant to the request, it will be proposed to the requesting agent as a partial (or total) solution. This is further discussed in Section 4.3.4. Otherwise, the agent replies with a message indicating a failure to meet the request.
4.3.4 Proposal Messages

Each proposal message contains a pre-commitment of the sender. A pre-commitment can be proposed as a deal to another agent or as a service offer in response to a previous request.

In the former case, the deal on the pre-commitment is relevant to some goals and/or base commitments of the owner agent and/or to some requirements of another proposal that it wants to fulfill. Assuming that the owner agent does not have prior knowledge to determine if the pre-commitment is likely to be accepted by a particular agent, the pre-commitment is then proposed to all agents. In the latter case, the pre-commitment is found by the sender to be relevant to the goal/base commitment of a previous request and hence proposed to the requesting agent as a response.

The relevance of a pre-commitment to a goal/base commitment is based on whether the pre-commitment’s post-conditions bring in some resources or make some actions available that fulfill partly or wholly the goal/base commitment. Further discussion on the relevance of a pre-commitment in Section 4.3.7.

Upon receiving a proposal, an agent also considers if the pre-commitment of the proposal is relevant to any requests previously sent or any goals (or commitments) of the recipient. If so, the recipient could further examine to see if the exchange of the pre-conditions for what it gains in the post-conditions is beneficial. However, we do not focus on such a mechanism to determine the benefits and hence assume that agents have such mechanisms in place and focus instead on the relevance criterion. Hence, if the pre-commitment is considered relevant, the recipient sends a message to indicate its acceptance of the proposal. Otherwise, it sends a message of rejection.

4.3.5 Fulfilling Conditions of Conditional Commitments

Once a proposal is accepted, its pre-commitment \((\text{time}_{\text{proposed}} \ [\Gamma \rightarrow \Delta])\) becomes a conditional commitment. This commitment is that if the recipient provides the pre-conditions within the required time frame, then the proposing agent commits to deriving \(\Delta\) at the corresponding time. In our framework, we only consider conditional commitments that are not broken. Hence, once the pre-conditions \(\text{time}_{\text{applied}} \ \Gamma\), which is also referred to as \conditions of the proposal, are fulfilled by the recipient, the proposing agent will definitely fulfill its commitment by applying the implication to transform the pre-conditions to the post-conditions and subsequently fulfill any base commitments in the post-conditions.

The time that the conditions of a proposal must be satisfied depends on the time frame
specified by \textit{time\_proposed}, and in many cases depends on the desired time of the post-conditions as well as the recipient’s choice.

For conditional commitments of the form $\bigcirc^n[\Gamma \rightarrow \Delta]$, the pre-conditions $\Gamma$ of the linear implication must be provided at the time $\bigcirc^n$ for the implication to be carried out. Hence, the conditions of the proposal are $\bigcirc^n\Gamma$. Note that being able to satisfy the conditions any time (i.e. deriving $\Box\Gamma$) is also acceptable because $\Box\Gamma \vdash \bigcirc^n\Gamma$.

For conditional commitments of the form $\Box[\Gamma \rightarrow \Delta]$, if the desired time of the post-conditions is $\bigcirc^n$, then the suitable time of the conditions is $\bigcirc^n$. If the desired time is $\Diamond$, then the time to satisfy the conditions should be either ”anytime” ($\Box$), ”sometime” ($\Diamond$), or a particular time ($\bigcirc^n$) of choice. If the desired time is $\Box$, then the time required of the conditions must be ”anytime” ($\Box$).

The application time of the linear implication is at a specific time point and is determined by the time point the conditions are satisfied or chosen in the case of anytime ($\Box$) or sometime ($\Diamond$).

The conditions of a conditional commitment can be attempted by the recipient as new sub-goals. Indeed, if what the recipient gains in the post-conditions can be regarded as new goals then the requirements to get them can be turned into sub-goals. These sub-goals can also be implemented as base commitments. However, unlike other base commitments, these base commitments can be aborted without forcing the interaction to fail. As noted, unfulfilled conditions leave the conditional commitment inactive. Moreover, as new sub-goals, the conditions can be attempted by the recipient using its own internal resources, actions and capabilities or using other agents’ via further interaction.

Once the conditions are fulfilled, the base commitments in the post-conditions become active and are subsequently attempted by the proposing agent.

\textbf{4.3.6 Fulfilling Goals and Base Commitments}

Base commitments may exist as a reason for interaction or as a result of some interactions. In the latter case, conditional commitments upon having their conditions fulfilled give rise to the base commitments specified in their post-conditions. We make the assumption that at the end of interaction, all the base commitments that result from the interaction must be fulfilled for the interaction to succeed. In other words, all the extra requirements as a result of the interaction must also be satisfied for the interaction to succeed.

To fulfill a base commitment, the necessary resources must be provided and/or the nec-
necessary actions must be carried out at the right time. These resources and actions can be provided within the owner agent (internal fulfillment) or alternatively provided by other agents in a deal (conditional commitment). In the latter case, the owner agent can propose pre-commitments that are relevant to the base commitments or may prefer to send requests of the base commitments to other agents. Correspondingly, a proposal or a request is made to another agent and the process for producing and handling a proposal or request to follow is as discussed above.

4.3.7 Utilizing Pre-commitments

Agents utilize relevant pre-commitments to answer requests from other agents and fulfill their own goals and base commitments. Indeed, the ability to select a pre-commitment that is relevant to a goal/base commitment is one of the key factors in our interaction model discussed above. We now look at the process of determining a relevant pre-commitment.

Relevance of a Pre-commitment to a Goal or Base Commitment

Intuitively, relevance here means that the use of the pre-commitment is beneficial to the fulfillment of the goal or base commitment. More precisely, the corresponding conditional commitment of the pre-commitment once fulfilled can produce the necessary resources and actions to fulfill part or all of the goal or base commitment. Correspondingly, (part of) formulas in the post-conditions of the pre-commitment can be used in a proof of (a part of) formulas of the goal or base commitment. Hence, relevance refers to the potential of a pre-commitment for fulfilling a goal or base commitment. This potential is determined by a number of factors as described below.

We firstly look at a fragment of TLL that is simple but powerful enough to examine the relevance relationship between a pre-commitment and a goal or base commitment.

Note that in our framework, capabilities are design objects, and agents do not have the ability to derive new capabilities. This has two important consequences. The first is that agents will not have any goals or base commitments to derive capabilities. Hence, formulas of goals or base commitments do not contain the connective $\rightarrow$. The second is that conditional commitments among agents do not produce any capabilities. Therefore, formulas on the right hand side of $\rightarrow$ in all pre-commitments and subsequently conditional commitments do not contain the $\rightarrow$ connective.

Furthermore, as requests contain only goals or base commitments, the request formulas
do not contain the $\rightarrow$ connective. Hence pre-commitments, whether being used to reply to a request or to fulfill a goal or base commitment, need to deal only with formulas that contain the connectives $\otimes$, $\&$, $\oplus$, $\square$, $\Diamond$, $\bigcirc$ and negation. Also, as will be discussed in Section 6.1, the modeling of the notions “anytime” and “sometime” can be made using only the connectives $\otimes$, $\&$, $\oplus$ and operator $\bigcirc$, which further reduces the logic fragment of concern to the one that contains only the connectives $\otimes$, $\&$, $\oplus$, $\bigcirc$ and negation.

Therefore, we can focus on discussing the relationship between a pre-commitment and a goal or base commitment in the context that formulas of goals or base commitments and post-conditions of pre-commitments contain only the connectives $\otimes$, $\&$, $\oplus$, $\bigcirc$ and negation.

As will be discussed in Section 6.2, this logic fragment facilitates the splitting up of formulars and techniques for handling them in parts. The parts can be as simple as basic TLL formulas, which are of the form $\bigcirc^i A$, where $A$ is an atomic LL formula (proposition). In particular, a formula $\Gamma$ that contains $\bigcirc^i A$ can be split up into two parts. One contains essentially $\bigcirc^i A$, possibly combined with constant 1, denoted as $\hat{A}$, while the other is the remainder which does not contain this $\bigcirc^i A$ (assuming that $\bigcirc^i A$ at different positions in the structure $\Gamma$ are different) and is denoted as $\Gamma A$. Formal definitions of $\Gamma A$ and $\hat{A}$ are provided in Definition 11 and Definition 13 respectively in Chapter 6.

Depending on the outcomes of internal choices and indeterminate possibilities that occur in $\hat{A}$, the formula $\bigcirc^i A$ may or may not eventually be retained. Note that these internal choices and indeterminate possibilities in $\hat{A}$ (and also $\Gamma A$) are taken from the formula $\Gamma$. Further discussions on the conditions for the $\bigcirc^i A$ to be retained in the part $\hat{A}$ are in Section 6.3.

We now consider two formulas $\alpha$ and $\beta$ in the fragment that contain a common basic TLL formula $\bigcirc^i A$. The two formulas $\alpha$ and $\beta$ can then be split up with respect to the common part. We examine the provability relationship between their split ups that contain $\bigcirc^i A$, denoted as $\hat{A}_\alpha$ and $\hat{A}_\beta$ respectively. If the retention of $\bigcirc^i A$ is made in both of them, they both become $\bigcirc^i A$ and hence one part can prove the other ($\bigcirc^i A \vdash \bigcirc^i A$). Otherwise, one part does not prove the other unless both of them become $\bigcirc^j 1$ for some $j$. Note that $\bigcirc^j 1$, in terms of resources and actions, does not contain anything relevant and hence is of no interest.

It can be seen that in determining if $\bigcirc^i A$ is retained, there is some risk associated with making assumptions about the outcomes of indeterminate possibilities. Whether to be bold or cautious with respect to certain indeterminate possibilities is a matter for the agents’ internal deliberation and hence beyond the scope of the thesis. We further discuss
a mechanism that reflects the agents’ strategies on determining choices and indeterminate possibilities in Chapter 5.

We introduce the concept of proof relevance as follows. Consider a formula $\alpha$ and a formula $\beta$ and their corresponding parts $\hat{A}_\alpha$ and $\hat{A}_\beta$ that contain a common TLL formula of the form $\Box A$. If the agent regards the part of $\alpha$ ($\hat{A}_\alpha$) proves the other part of $\beta$ ($\hat{A}_\beta$) then the formula $\alpha$ is considered as proof relevant to the formula $\beta$.

**Definition 1.** Proof Relevance

Let $\Gamma$ and $\Delta$ be TLL formulas that have a common sub-formula $A$. Let $\hat{A}_\Gamma$ and $\hat{A}_\Delta$ be two split ups of $\Gamma$ and $\Delta$ respectively regarding $A$ (as defined in Definition 13 in Chapter 6). If the agent regards $\hat{A}_\Gamma \vdash \hat{A}_\Delta$, then $\Gamma$ is said to be proof relevant to $\Delta$.

Note that although “the agent regards $\hat{A}_\Gamma \vdash \hat{A}_\Delta$”, it may later turn out that $\hat{A}_\Gamma \not\vdash \hat{A}_\Delta$.

Also, the notion of proof relevance can be implemented by a procedure that relies on proof search (for $\hat{A}_\Gamma \vdash \hat{A}_\Delta$) under a particular strategy on choices and indeterminate possibilities that are involved in the proof search. The key to implementing such procedure is choice calculus, which is discussed in Chapter 5. In particular, techniques for conducting proof search following a particular strategy can make use of the choice calculus sequent rules. It is important to note that checking for proof relevance between two formulas does not require any non-logical axioms to be used. To some extent, searching for a proof in this context can be understood as a unification process between formulas on the right of $\vdash$ and those on the left. Potential techniques for choice calculus may be derived partly from existing techniques in linear logic, which are discussed in Section 2.5.2. Developing a proof search technique for the full fragment of TLL is however outside the scope of this thesis.

Based on the notion of proof relevance, a pre-commitment is regarded as relevant to a goal/base commitment if the formula of the post-conditions is proof relevant to the formula of the goal/base commitment.

It can be seen that whether a pre-commitment is relevant or not depends on the agent’s strategies with respect to the internal choices and the degree of risk taking in assuming the outcomes of the indeterminate possibilities involved. In particular, if the agent does not want to take risks, there must be a proof of (a part of) the goal or base commitment formula using (a part of) the post-conditions formula for all the possible outcomes of any indeterminate possibilities involved. If the agent is willing to take risks, it allows some uncertainty in its assumptions on the outcomes of indeterminate possibilities. If the assumptions turn out to be wrong, then this also means that the pre-commitment turns out to be not relevant. In
other words, relevance does not always imply that the pre-commitment will be used (partly) in proving (part of) the goal or base commitment, but reflects that the agent is taking a chance that it might be so.

In addition, a pre-commitment may fulfill one part of a goal or base commitment or a request while the other parts of the goal or base commitment or request are fulfilled by other pre-commitments. The fulfillment of a goal or base commitment or request can hence be regarded as a form of distributed concurrent problem solving with a dynamic arrangement of tasks (handling pre-commitments) among multiple players. Further discussion about distributed concurrent problem solving is in Section 6.5.

4.3.8 Integration of Changes into Protocol Execution

Since agents operate in a multi-agent environment, they have to deal with not only changes from the environment but also changes introduced by other agents. Also, agents are likely to participate in multiple interactions with other agents and these interactions can affect each other. Although changes might come from such different sources, in our interaction framework, these changes manifest themselves by the changes occurring to the agents’ states. In other words, the states of the agents are the grounds for various sources to interact. These changes include those about resources, system properties, and commitments.

Moreover, changes might also occur to the set of pre-commitments of each agent as new pre-commitments could be formed or existing ones merged or split up during interaction. Though it is also important to investigate how protocols evolve through changes in the agents’ pre-commitments, we assume that pre-commitments are designed beforehand by humans and do not change during the course of interaction.

Hence, we examine the effects of on-going changes from the environment and other agents on the agents’ resources, properties, base commitments and conditional commitments in their states. Changes are introduced to the agents’ states by the addition and removal of formulas.

By taking into account the agents’ current states, the interaction process can reflect recent changes and adapt execution to take advantage of opportunities or deal with exceptions introduced by these changes. Specifically, in our framework, agents construct protocols based around specified pre-commitments. Changes in state may then remove some inputs and/or make others available, hence enabling and disabling the corresponding pre-commitments. State changes may also make some (part of) goals or base commitments fulfilled or some conditional commitments active or even introduce new goals or base commitments. In effect,
by reasoning about changes in state formulas, agents guide the interaction in appropriate ways.

Consider our cricket bat sale example. The addition of two junior cricket bats to the customer’s state as a result of another interaction thread allows the customer and merchant to skip the interaction. Another possibility is for the merchant to already know the customer credit card number (perhaps retained from a previous transaction) and therefore their interaction can bypass the part that involves sending the credit card number to the merchant.

4.3.9 Cricket Bat Sale Interaction

Based on the specification of the cricket bat sale example as described in Section 3.2.5 and Section 4.2.4, we examine how the interaction model in Section 4.3.1 is deployed to construct interaction sequences among agents. An example of such interaction is as follows.

For simplicity, we make use of the following shortcut rules. Note that \([n]\) in the title of a rule denotes multiple times of applications of that rule.

\[
\frac{\Gamma, \Box^n F \vdash \Delta}{\Gamma, \Box F \vdash \Delta} \quad \Box L
\]

for some \(n\) is a shortcut for

\[
\frac{F \vdash F \quad \Box F \vdash F}{\Box F \vdash \Box F} \quad \Box L
\]

\[
\frac{\Box F \vdash \Box^n F \quad \Gamma, \Box^n F \vdash \Delta}{\Gamma, \Box^n F \vdash \Delta} \quad \text{cut}
\]

\[
\frac{\Gamma \vdash \Box^n F, \Delta}{\Gamma \vdash \Diamond F, \Delta} \quad \Diamond L
\]

for some \(n\) is a shortcut for

\[
\frac{F \vdash F \quad \Diamond R}{\Diamond F \vdash \Diamond F} \quad \Diamond R
\]

\[
\frac{\Box^n F \vdash \Diamond F \quad \Gamma \vdash \Box^n F, \Delta}{\Gamma \vdash \Diamond F, \Delta} \quad \text{cut}
\]

\[
\frac{\Gamma, \Box^n (F \otimes G) \vdash \Delta}{\Gamma, \Box^n F \otimes \Box^n G \vdash \Delta} \quad \Box (\otimes)L
\]

\[
\frac{\Gamma, \Box^n F \vdash \Delta}{\Gamma, \Box^n F \otimes \Box^n G \vdash \Delta} \quad \Box (\otimes)L
\]
for some $n$ is a shortcut for

$$
\frac{F \vdash F \quad G \vdash G}{F,G \vdash F \otimes G} \otimes R
$$

$$
\frac{\bigcirc^n F, \bigcirc^n G \vdash \bigcirc^n (F \otimes G)}{\Gamma, \bigcirc^n F, \bigcirc^n G \vdash \Delta} \bigcirc L
$$

$$
\frac{\Gamma \vdash \bigcirc^n F \otimes \bigcirc^n G, \Delta}{\Gamma \vdash \bigcirc^n (F \otimes G), \Delta} \bigcirc (\otimes) R
$$

for some $n$ is a shortcut for

$$
\frac{F \vdash F \quad G \vdash G}{F,G \vdash F \otimes G} \otimes R
$$

$$
\frac{\bigcirc^n F, \bigcirc^n G \vdash \bigcirc^n (F \otimes G)}{\bigcirc^n F \otimes \bigcirc^n G \vdash \bigcirc^n (F \otimes G)} \otimes L
$$

$$
\frac{\Gamma \vdash \bigcirc^n F \otimes \bigcirc^n G, \Delta}{\Gamma \vdash \bigcirc^n (F \otimes G), \Delta} \bigcirc L
$$

$$
\vdash \bigcirc^n \Gamma
$$

$$
\square [\Gamma \rightarrow \Delta] \vdash \bigcirc^n \Delta" app
$$

for some $n$ is a shortcut for

$$
\frac{\Gamma \vdash \Gamma \quad \Delta \vdash \Delta}{\Gamma \otimes [\Gamma \rightarrow \Delta] \vdash \Delta} \rightarrow L
$$

$$
\frac{\bigcirc^n (\Gamma \otimes [\Gamma \rightarrow \Delta]) \vdash \bigcirc^n \Delta}{\bigcirc^n \Gamma \otimes \bigcirc^n [\Gamma \rightarrow \Delta] \vdash \bigcirc^n \Delta} \bigcirc (\otimes) L
$$

$$
\vdash \bigcirc^n \Gamma
$$

$$
\frac{\bigcirc^n \Gamma \otimes \square [\Gamma \rightarrow \Delta] \vdash \bigcirc^n \Delta}{\square [\Gamma \rightarrow \Delta] \vdash \bigcirc^n \Delta} cut
$$

Moreover, in the illustration, we will make use of a splitting mechanism of a compound formula which will be discussed in Section 6.2. For example, a conjunct $\bigcirc^n A$ can be extracted from a multiplicative conjunction $\bigcirc^n (A \otimes B \otimes C)$. Hence, by saying “extract” a formula from a compound formula, we refer to the handling of splitting up formulas in Section 6.2 which is not discussed in detail here.

A summary of specifications of the cricket bat sale example is given in Table 4.1. The pre-commitments and capabilities of participating agents are both named as rules. Also, the pre-commitments and capabilities are given without using variables but the actual agent names
to indicate locations and ownerships of resources and actions. This is intended to simplify the illustration. However, the illustrated interactions can be extended using variables for locations and ownerships by simply adding the assignment of the variables as appropriate. In addition, resources and actions are shown together under category named “ResAct”.

<table>
<thead>
<tr>
<th>Mer</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResAct</td>
</tr>
<tr>
<td>Rule 1</td>
</tr>
</tbody>
</table>

\[
\eta \square \left[10 \square \text{Cus} \subseteq (\text{Mer} \otimes \text{Ebank}) \otimes \text{addr}@C_c \right] \\
\rightarrow \left[10 \square \text{jr}_c \otimes \text{Cus} \subseteq \text{Mer} \otimes \text{Ebank} \otimes \text{addr}@C_c \right]
\]

| ResAct |
| Rule 2 |

\[
\eta \square \left[(20 \square \text{Cus} \subseteq (\text{Mer} \otimes \text{Ebank}) \otimes \text{addr}@C_c) \right] \\
\rightarrow \left[20 \square \text{Cus} \subseteq \text{Mer} \otimes \text{Ebank} \otimes \text{addr}@C_c \right]
\]

| ResAct |
| Rule 3 |

\[
\eta \square \left[\text{quote}@B_m \subseteq \text{Cus} \subseteq \text{Mer} \otimes \text{addr}@C_c \right] \\
\rightarrow \left[\text{quote}@B_m \subseteq \text{Cus} \subseteq \text{Mer} \otimes \text{addr}@C_c \right]
\]

| Goal |
| Rule 6 |

\[
\eta \square \left[(\text{Mer} \otimes \text{Ebank}) \subseteq \text{Cus} \subseteq \text{Mer} \otimes \text{addr}@C_c \right] \\
\rightarrow \left[\text{Mer} \otimes \text{Ebank} \subseteq \text{Cus} \subseteq \text{Mer} \otimes \text{addr}@C_c \right]
\]

\[
\text{Table 4.1: Summary of Specification for Cricket Bat Sale Example}
\]

The customer Cus has a goal of having two junior cricket bats and a gift and cannot fulfill the goal by itself. The interaction can start by a request from the customer Cus of its goal to other agents.

1. Cus sends a request for two junior cricket bats to be delivered at Cus and a gift at some time to other agents. Note that requests to agents other than Mer (for example agent O) will result in a failure:

\[
\text{C to O: REQUEST } [\Diamond (2 \text{jr}_c @ C_c \otimes \text{gift}@C_c)]^\perp \\
\text{X to O: REQUEST } [\Diamond (2 \text{jr}_c @ C_c \otimes \text{gift}@C_c)]^\perp \text{ FAILS.}
\]

For simplicity, we will only show the requests that go to the correct agents. Consider a request to agent Mer

\[
\text{C to M: REQUEST } [\Diamond (2 \text{jr}_c @ C_c \otimes \text{gift}@C_c)]^\perp
\]
Mer analyzes the request.

\[\vdash 2jr\_cb@C_c \vdash gift@C_c \otimes R\]
\[\vdash 2jr\_cb@C_c \otimes gift@C_c \otimes R\]
\[\vdash \diamondsuit (2jr\_cb@C_c \otimes gift@C_c)\]
\[\vdash \diamondsuit (2jr\_cb@C_c \otimes gift@C_c) \vdash \downarrow R \vdash (1\ast) \vdash\]

Hence, to meet the request, Mer looks for rules that will allow it to derive

\[\circ^n 2jr\_cb@C_c, \text{ and } \circ^n gift@C_c.\]

For \[\circ^n 2jr\_cb@C_c\], there are two applicable rules - rule 1 and rule 2.

Two applications of rule 1 at the \(n\)th next time point can derive \[\circ^n 2jr\_cb@C_c\]. Denote rule 1 as \(\Box[L1 \rightarrow R1]\). An application of rule 1 at the \(n\)th next time point is as follows:

\[\vdash \circ^n L1\]
\[\Box[L1 \rightarrow R1] \vdash \circ^n R1 \text{ app}\]

Given that \[\circ^n R1\] can be derived from rule 1 where

\[\circ^n R1 = \circ^n [10 \Box \$@M \otimes \Box jr\_cb@C_c \otimes jr\_cb@M \otimes \Box addr@M_c]\]

at the \((n)\)th next time point, we can extract \[\circ^n \Box jr\_cb@C_c\] and make the derivation

\[\circ^n \Box jr\_cb@C_c \vdash \circ^n jr\_cb@C_c.\]

Regarding rule 2, an application of the rule at the \(n\)th next time point can derive \[\circ^n 2jr\_cb@C_c\] and \[\circ^n gift@C_c\]. Denoting rule 2 as \(\Box[L2 \rightarrow R2]\), we have

\[\vdash \circ^n L2\]
\[\Box[L2 \rightarrow R2] \vdash \circ^n R2 \text{ app}\]
\[\vdash (2\ast) \vdash\]

Given that

\[\circ^n R2 = \circ^n [20 \Box \$@M \otimes \Box 2jr\_cb@C_c \otimes \Box gift@C_c \otimes 2jr\_cb@M \otimes gift@M \otimes \Box addr@M_c]\]

at the \((n)\)th next time point, we can extract \[\circ^n \Box 2jr\_cb@C_c\], and \[\circ^n \Box gift@C_c\] and make the derivations
According to (1*), rule 2 can derive all of the Cus’ request while rule 1 can derive only 2 junior cricket bats without a gift. Hence, rule 2 will be chosen to answer the Cus’ request.

2. Mer proposes to Cus that at the \((n)^{th}\) next time point, Mer will commit to giving 2 junior cricket bats and a gift to Cus if Cus pays 20 dollars either via Paypal or by credit card and gives Mer its address.

\[
\text{M to C: PROPOSE } \Box^n [20\$ @ C \otimes (\text{via PP} @ M_c \oplus \text{by cred} @ M_c) \otimes \text{addr} @ C_c] \\
\rightarrow 20 \square \$ @ M_M \otimes \Box^n 2 \text{ jr cb} @ C_c \otimes \Box \text{gift} @ C_c \otimes 2 \text{ jr cb} @ M_M^\perp \otimes \Box \text{addr} @ M_c]
\]

With similar reasoning as (1*), Cus determines that the proposal can help to derive its request. Hence

\[
\text{C to M: ACCEPT}
\]

The conditions of this proposal are those that enable the proposal to be applied. According to (2*), they are:

\[
\Box^n L2 = \Box^n [20\$ @ C_c \otimes (\text{via PP} @ M_c \oplus \text{by cred} @ M_c) \otimes \text{addr} @ C_c].
\]

Cus analyzes the conditions:

\[
\begin{array}{c}
\vdash \text{via PP} @ M_c \text{ or } \vdash \text{by cred} @ M_c \\
\vdash 20\$ @ C_c \vdash \text{addr} @ C_c \\
\vdash 20\$ @ C_c \otimes (\text{via PP} @ M_c \oplus \text{by cred} @ M_c) \otimes \text{addr} @ C_c \oplus R \\
\vdash \Box^n [20\$ @ C_c \otimes (\text{via PP} @ M_c \oplus \text{by cred} @ M_c) \otimes \text{addr} @ C_c] \text{ n } \Box
\end{array}
\]

From (3*), one way to satisfy the conditions is that at the \(n^{th}\) next time point, Cus will reserve 20 dollars, send its address to Mer, and carry out payment either via Paypal or by credit card. The choice is decided by Cus.

3. To derive \(\Box^n 20\$ @ C_c\), Cus uses 20 dollars of its resources.

\[
20 \square \$ @ C \vdash \Box 20\$ @ C \vdash \Box^n 20\$ @ C.
\]

4. To derive \(\Box^n \text{addr} @ C_c\), Cus uses its address as
\( \Box \text{addr} \otimes C_c \vdash \Diamond^n \text{addr} \otimes C_c \).

Note that an application of the rule as in (2\*) will consume \( \Diamond^n \text{addr} \otimes C_c \) to derive \( \Box \text{addr} \otimes M_c \), which means sending Cus’s address to Mer.

5A. Deriving \( \Diamond^n \text{by.cred} \otimes M_c \) means that Cus chooses to pay by credit card to Mer at the \( n^{th} \) next time point.

To derive \( \Diamond^n \text{by.cred} \otimes M_c \), Cus can make use of rule 6. Denoting rule 6 as \( \Box [L6 \rightarrow R6] \), an application of rule 6 at the \( n^{th} \) next time point is as follows:

\[
\vdash \Diamond^n L6 \quad \Box [L6 \rightarrow R6] \vdash \Diamond^n R6 \quad \text{app}
\]

where \( \Diamond^n R6 = \Diamond^n \Box \text{by.cred} \otimes M_c \) and

\[
\Diamond^n \Box \text{by.cred} \otimes M_c \vdash \Diamond^n \text{by.cred} \otimes M_c.
\]

The conditions for the application of the rule are

\[
\Diamond^n L6 = \Diamond^n [(\text{cred.no} \otimes C_c \rightarrow \text{cred.no} \otimes M_c)^{\perp} \otimes \text{cred.paym} \otimes M_c].
\]

Cus analyzes the conditions:

\[
\begin{align*}
\vdash (\text{cred.no} \otimes C_c \rightarrow \text{cred.no} \otimes M_c) & \vdash (\text{cred.no} \otimes C_c \rightarrow \text{cred.no} \otimes M_c)^{\perp} \rightarrow L \quad \vdash \text{cred.paym} \otimes M_c \rightarrow R \\
\vdash (\text{cred.no} \otimes C_c \rightarrow \text{cred.no} \otimes M_c)^{\perp} \otimes \text{cred.paym} \otimes M_c & \vdash \Diamond^n [(\text{cred.no} \otimes C_c \rightarrow \text{cred.no} \otimes M_c)^{\perp} \otimes \text{cred.paym} \otimes M_c]^{\perp} \rightarrow R
\end{align*}
\]

Hence, \( \Diamond^n (\text{cred.no} \otimes C_c \rightarrow \text{cred.no} \otimes M_c) \) will be inserted into Cus’s state formulas and Cus has to derive \( \Diamond^n \text{cred.paym} \otimes M_c \).

Denoting \( \text{cred.n} \otimes C_c \) as \( \text{crC} \) and \( \text{cred.no} \otimes M_c \) as \( \text{crM} \), we have

\[
\begin{align*}
\vdash \text{crM} \otimes \text{crC} & \rightarrow \text{crM} \rightarrow \text{crC} \rightarrow \text{crM} \rightarrow \text{crC} \rightarrow L \\
\vdash \Diamond^n (\text{crM} \otimes \text{crC} \rightarrow \text{crM}) \rightarrow \Diamond^n \text{crC} \rightarrow L \\
\vdash \Diamond^n (\text{crC} \rightarrow \text{crM}) \rightarrow \Diamond^n \text{crC} \rightarrow \text{cut}
\end{align*}
\]

where \( \Diamond^n \text{cred.no} \otimes C_c \) is further required to obtain \( \Diamond^n \text{cred.no} \otimes M_c \), which means sending Mer Cus’ credit card number at the \( n^{th} \) time point. Given that Cus has \( \Box \text{cred.no} \otimes C_c \), we have
From Specification to Execution

\[ \Box cred\_no@C_C \vdash \Diamond^n cred\_no@C_C. \]  
\[ \neg (4*) \]

6A. Because Cus does not have any rule to derive \( cred\_paym@M_M \), it will send a request to another agent (Mer).

C to M: REQUEST \( \Diamond^n cred\_paym@M_M \)

In order to answer the request, rule 3 can be used. Denoting rule 3 as \( \Box [L3 \rightarrow R3] \), then

\[ \vdash \Diamond^n L3 \]
\[ \Box [L3 \rightarrow R3] \vdash \Diamond^n R3 \]

where \( \Diamond^n R3 \) can derive \( \Diamond^n cred\_paym@M_M \) as

\[ \Diamond^n \Box cred\_paym@M_M \vdash \Diamond^n cred\_paym@M_M. \]

The conditions are then

\[ \Diamond^n L3 = \Diamond^n [(cred\_no@M_C \rightarrow cred\_no@B_C) \perp \otimes cred\_paym@M_B]. \]

Mer analyzes the conditions:

\[ \vdash (cred\_no@M_C \rightarrow cred\_no@B_C) \]
\[ \vdash (cred\_no@M_C \rightarrow cred\_no@B_C) \perp \rightarrow L \]
\[ \vdash cred\_paym@M_B \otimes R \]
\[ \vdash \Diamond^n [(cred\_no@M_C \rightarrow cred\_no@B_C) \perp \otimes cred\_paym@M_B] \]
\[ \Diamond^n \]

Hence, \( \Diamond^n (cred\_no@M_C \rightarrow cred\_no@B_C) \) will be inserted into Mer’s state formulas and Mer has to derive \( \Diamond^n cred\_paym@M_B \).

With similar reasoning to \( (4*) \), Mer can use

\[ \Diamond^n (cred\_no@M_C \rightarrow cred\_no@B_C) \] and \( \Diamond^n cred\_no@M_C \)

to derive \( \Diamond^n cred\_no@B_C \), which means sending Cus’s credit card number to EBank at the \( n^{th} \) time point.

7A. Because Mer does not have any rule to derive \( \Diamond^n cred\_paym@M_B \), it will make a request to another agent.

Consider a request to Ebank.

M to B: REQUEST \( \Diamond^n cred\_paym@M_B \)
An application of rule 5 (denoted as $\square[L5 \rightarrow R5]$) may satisfy the request.

$$\vdash \Diamond^n L5$$

$$\square[L5 \rightarrow R5] \vdash \Diamond^n R5 \text{ app}$$

where $\Diamond^n R5 = \Diamond^n [\square cred\_paym@M_B \oplus \square cred\_disappr@M_B]$. $\Diamond^n (\square cred\_paym@M_B \oplus 1)$ can be extracted from $\Diamond^n R5$ and may turn out to be $\Diamond^n \square cred\_paym@M_B$ and

$$\Diamond^n \square cred\_paym@M_B \vdash \Diamond^n cred\_paym@M_B.$$ 

Hence, Ebank proposes rule 5 to Mer at the $n^{th}$ time point. If Mer provides a quote of sale and Cus’s credit number to Ebank, Ebank commits that it will arrange a credit payment (if Cus’s credit is approved) or send back its disapproval of Cus’s credit (otherwise). There is also a requirement of sending the quote of sale to EBank ($quote\_req@M_B$) given to Mer.

$B$ to $M$: PROPOSE $\Diamond^n[quote@B_M \otimes cred\_no@B_C \otimes quote\_req@M_B] \rightarrow$

$$\square cred\_paym@M_B \oplus \square cred\_disappr@M_B$$

With similar reasoning on the suitability of the proposal to the request, Mer will accept the proposal.

$M$ to $B$: ACCEPT

The conditions are

$$\Diamond^n L5 = \Diamond^n (quote@B_M \otimes cred\_no@B_C \otimes quote\_req@M_B).$$

Mer analyzes the condition at the $n^{th}$ time point:

$$\vdash quote@B_M \vdash cred\_no@B_C \vdash quote\_req@M_B \rightarrow L$$

$$\vdash quote@B_M \otimes cred\_no@B_C \otimes quote\_req@M_B \rightarrow R$$

$$\vdash \Diamond^n [quote@B_M \otimes cred\_no@B_C \otimes quote\_req@M_B]$$

$\square \Diamond^n$.

Hence, $\Diamond^n quote\_req@M_B$ (a requirement of a quote of sale from EBank) will be inserted into Mer’s state formulas and together with Mer’s resources, make the conditions of rule 4 satisfied. Denoting rule 4 as $\square[L4 \rightarrow R4]$, then an application of rule 4 is as follows

$$\vdash \Diamond^n L4$$

$$\square[L4 \rightarrow R4] \vdash \Diamond^n R4 \text{ app}$$
where $\Diamond^n R4$ derives $\Diamond^n \text{quote} @ B_M$ as

$$
\Diamond^n \square \text{quote} @ B_M \vdash \Diamond^n \text{quote} @ B_M.
$$

The conditions of the application of rule 4 are then

$$
\Diamond^n LA = \Diamond^n \left[\text{issue\_quote} @ M \otimes \text{quote\_req} @ M_B\right]
$$

which are analyzed as

$$
\vdash \text{issue\_quote} @ M \vdash \text{quote\_req} @ M_B \otimes R \\
\vdash \text{issue\_quote} @ M \otimes \text{quote\_req} @ M_B \vdash \Diamond^n \left[\text{issue\_quote} @ M \otimes \text{quote\_req} @ M_B\right] n \Diamond
$$

where Mer has $\Diamond^n \text{quote\_req} @ M_B$ and $\Diamond^n \text{issue\_quote} @ M$ is obtained from its resources as $\square \text{issue\_quote} @ M \vdash \Diamond^n \text{issue\_quote} @ M$.

Hence, rule 4 can be applied to derive $\Diamond^n \text{quote} @ B_M$. Moreover, from the end result in part 6A, $\Diamond^n \text{cred\_no} @ B_C$ is also given. Together, they satisfy the conditions of the application of rule 5 $\Diamond^n L5$.

Ebank can then apply the proposal or rule 5 to derive

$$
\Diamond^n (\square \text{cred\_paym} @ M_B \oplus \square \text{cred\_disappr} @ M_B).
$$

As $\oplus$ indicates the choice is external to both agents. There are two cases.

**Case 1:** at the $n^{th}$ time point, Cus’s credit is approved by Ebank. Then

$$
\Diamond^n (\square \text{cred\_paym} @ M_B \oplus \square \text{cred\_disappr} @ M_B)
$$

becomes

$$
\Diamond^n \square \text{cred\_paym} @ M_B \vdash \Diamond^n \text{cred\_paym} @ M_B.
$$

As a result, at the $(n)^{th}$ next time point, Ebank will arrange the credit payment, fulfilling the Mer’s request for $\Diamond^n \text{cred\_paym} @ M_B$.

**Case 2:** at the $n^{th}$ time point, Cus’s credit is not approved. Then

$$
\Diamond^n (\square \text{cred\_paym} @ M_B \oplus \square \text{cred\_disappr} @ M_B)
$$

becomes
\(\Box^n \square cred\_disappr@M_B \vdash \Box^n\text{cred\_disappr}@M_B\)

EBank’s attempt at the Mer’s request for \(\Box^n\text{cred\_paym}@M_B\) fails.

\[\text{B to M: REQUEST } [\Box^n\text{cred\_paym}@M_B]^\perp \text{ FAILS}\]

As a result, Mer can not apply rule (3) and so fails to answer the Cus’s request for \([\Box^n\text{cred\_paym}@M_M]^\perp\) in part 6A.

\[\text{M to C: REQUEST } [\Box^n\text{cred\_paym}@M_M]^\perp \text{ FAILS}\]

A lack of \(\Box^n\text{cred\_paym}@M_M\) means that rule 6 at Cus can not be applied and Cus fails to derive \(\Box^n\text{by\_cred}@M_C\). Cus may backtrack to take the option to pay via Paypal.

5B. Deriving \(\Box^n\text{via\_PP}@M_C\) means that Cus carries out Paypal payment method to pay Mer at the \((n)\)th next time point. This happens when Cus chooses to pay via Paypal initially or as a result of backtracking from a failure in credit payment.

As Cus can carry out the action \(\square\text{via\_PP}@M_C\), deriving \(\Box^n\text{via\_PP}@M_C\) can be done:

\(\square\text{via\_PP}@M_C \vdash \Box^n\text{via\_PP}@M_C\).

8. Mer fulfills Cus’ initial request.

When any of \(\Box^n\text{via\_PP}@M_C\) or \(\Box^n\text{by\_cred}@M_C\) is derived, together with the other conditions (\(\Box^n\text{20}\$@C_C\), \(\Box^n\text{addr}@C_C\)) being satisfied, they allow Mer to apply the initial proposal of rule 1 to derive \(\Box^n\text{R1}\). From \(\Box^n\text{R1}\), \(\Box^n \square \text{gift}@C_C\) and \(\Box^n \square \text{jr\_cb}@C_C\) can be extracted and because

\(\Box^n \square \text{gift}@C_C \vdash \Box^n\text{gift}@C_C\) and
\(\Box^n \square \text{jr\_cb}@C_C \vdash \Box^n\text{jr\_cb}@C_C\),

according to \((1^*)\), Cus’s initial request is satisfied.

The interaction ends successfully as Cus’s initial request is satisfied and there are no pending commitments generated by the interaction.

The interaction has achieved the flexibility desired based on the encoding. In fact, \(\Diamond\) in \(\Diamond(2\text{jr\_cb}@C_M \otimes \text{gift}@C_M)\) allows Mer to have a time of its choice \((n)\) to do the sale. Also, a choice of which \text{via\_PP}@M_C (Paypal payment) or \text{by\_cred}@M_C (credit payment) will be derived can be decided by Cus. Similarly, the indeterminate possibility in the decision on credit payment (step 7A of the interaction) is also captured and for all the cases that might
happen, the merchant and customer have to deal with. In particular, when Cus’s credit payment can not be arranged, the customer has to backtrack to the point of making the choice of payment to take another option, which is reflected in the interaction.

Moreover, those attempts to derive 20 dollars ($^{20}\text{C}_C$), to send the address ($^{\text{addr}}\text{M}_C$), to reserve 2 junior cricket bats ($^{2\text{jr}_{cb}}\text{M}_M$) and to arrange a payment method ($^{\text{via}_{PP}}\text{M}_C$ or $^{\text{by}_{cred}}\text{M}_C$) can occur in any order. This also adds to the flexibility of the interaction.

In all, this section has demonstrated how an interaction might be constructed from a specification in our framework based on our interaction model and by using proof search techniques.

4.4 Summary

The chapter described how protocols and interaction can be specified for a resource-conscious commitment-based agent. We also discussed several related issues involved with the goals or base commitments of participating agents. We also discussed a possible execution framework to realize such TLL specifications which enables agents to negotiate commitments among themselves and utilizing resources, actions, capabilities and pre-commitments to fulfill their goals and commitments. In the next chapter, we will explore a logic mechanism that can be used to enable agents to reason about their decisions about choices.
Chapter 5

Modeling Decisions about Choices

In Chapter 3, we have presented a model of resource and commitment-based interaction in which internal choices and indeterminate possibilities are modeled simply as relationship between resources, actions, capabilities, goals and commitments. In Chapter 4, a specification framework for interaction protocols is presented based around the central notion of pre-commitments. We also introduced an interaction model to realize specifications based on utilizing pre-commitments to fulfill goal or base commitments of agents. Another important part of how an agent utilizes its resources, actions, capabilities and pre-commitments in interaction is making decisions on internal choices and handling indeterminate possibilities.

Decisions on internal choices made now may affect the future achievement of goals or base commitments or other threads of interaction. Agents should be enabled to make informed and sensible decisions about choices. In open and dynamic operating environments, changes from the environment occur frequently and often are unpredictable, which can hinder the accomplishment of the agents’ goals. How agents cope with changes, which are reflected in indeterminate possibilities, remains an open and challenging problem. On the one hand, agents should be enabled to reason about the current changes and act flexibly. On the other hand, agents should be equipped with a reasoning ability to predict changes and act accordingly.

These characteristics are desirable for a single agent. However, an agents’ decisions are not made in isolation, but in the context of decisions made by other agents, and as part of interactions between the agents. Thus, the challenging setting here is that in negotiation and other forms of agent interaction, decision making is distributed. In particular, the key challenges are:
- **Distribution**: choices are distributed among agents, and changes from the environments affect each agent in different ways. It is vital to capture these choices, their dependencies and the effects of different agent strategies on them.

- **Time**: internal choices may be time-bounded and changes in the environment occur in time. Therefore, decisions on them should be made in a time-dependent manner.

- **Dependencies**: i.e. capturing that certain decisions depend on other decisions.

In order to address these challenges, one approach is using formalisms such as TLL to enable agent reasoning about choices and indeterminate possibilities. TLL is highly suitable here because it allows us to use time-dependent contexts.

Given that the current state of an agent can be modeled as a collection of formulas, consequences of a particular decision on an internal choice or an indeterminate possibility can be explored via standard reasoning methods. Hence, we model agents’ decisions about choices and indeterminate possibilities in a way that allows distribution, dependencies, and time information to be captured. We discuss specific desirable properties of a formal model of agent decisions on choices and choice calculus in Section 5.1 and then present the formal model in Section 5.2 and the choice calculus in Section 5.3.

### 5.1 Desiderata for a Choice Calculus

Many unpredictable changes in the environment can be regarded as a set of possibilities for which the agents do not know the outcomes. There are several strategies for dealing with unpredictable changes. A safe approach is to prepare for all the possible scenarios, which is at the cost of extra reservation and/or consumption of resources. Other approaches are more risky in which agents make predictions about which possibilities will occur and act accordingly. If the predictions are correct, agents achieve their goals more efficiently. Underlying these varying approaches is a trade-off between resource efficiency and safety. Hence, it is important to model their predictions on the indeterminate possibilities of concern and allow them to reason about the consequences.

In contrast to indeterminate possibilities, internal choices are what agents can decide by themselves. Decisions about choices can be based on what is best for the agents’ current and local needs. However, it is desirable that they consider the choices in the context of the other choices that have been and/or will be made to ensure the overall consistency. This requires an ability to make an informed decision on choices. If we record information
about internal choices as constraints associated with these choices then what is required is a model of internal choices with their associated constraints. This model will allow agents to reason about internal choices and decide accordingly. Also, in a distributed environment like a multi-agent system, this modeling should take into account further constraints such as dependencies between choices.

In addition, as agents act in time, decisions can be made at the required time or can be made in advance. In some cases, agents may decide an internal choice in advance to save resources. For example, consider a goal \( \bigcirc^3(A \oplus B) \). This goal involves an internal choice \((\oplus)\) to be determined at the third next time point \((\bigcirc^3)\). If the agent decides now to choose \(A\) and keeps that decision the same at the third next time point, then from now on, the agent only has to prepare resources for the goal of \(\bigcirc^3A\). This also means that other resources can now be guaranteed to be exclusive from the requirements of \(\bigcirc^3(A \oplus B)\) and can be used to achieve other goals. This might not be the case if later \(\bigcirc^3(A \oplus B)\) is decided as \(\bigcirc^3B\). Hence, it is important to model the ability of agents to make decisions on internal choices and indeterminate possibilities in advance and remain consistent.

As a running example, we consider the following scenario which illustrates various desirable strategies of agents on their decisions over internal choices and indeterminate possibilities.

*Peter intends to organize an outdoor party in the next two days. He has a goal of providing music at the party. He also has a blank CD which he can use with his CD burner to burn music in CD format or mp3. His friend, John, can help by bringing a CD player or an mp3 player to the party but Peter does not know which until tomorrow. In addition, there is an external request that David wants to borrow Peter’s CD burner today. Peter needs to consider achieving his goal and whether to let David borrow the CD burner.**

In this situation, there is an internal choice on the music format to be made by Peter and also an indeterminate possibility regarding the player. An encoding in TLL of the example is as follows. To focus on discussion about choices, for simplicity, we omit information on location and ownership. We also make some abbreviations for the purpose of presenting proofs. Note that the numbers in subscripts of the connectives & and \(\oplus\) indicate IDs of the respective choices.

Peter’s goal is to have music two days later

\[(\bigcirc^3m)^\perp\]
Desiderata for a Choice Calculus

Peter’s resources (denoted as PR): a blank CD and a CD burner ($\text{CDb}$)

$$\Box \text{CD} \otimes \Box \text{CDb}.$$  

Peter’s capability of burning music to CD (denoted as PB for "Burn"):

$$\Box [(\text{CD} \otimes \text{CDb}) \rightarrow (\Box \text{mp3} \otimes_1 \Box \text{CDf})]$$

i.e. at any time, Peter can convert a blank CD to either an mp3 or music format CD ($\text{CDf}$). The choice $\otimes_1$ is an internal choice.

Peter’s capability of playing music (denoted as PP for "Play"):

$$\Box [[[\text{mp3} \otimes \text{mp3}] \oplus_2 (\text{CDf} \otimes \text{CDp})] \rightarrow m]$$

i.e. at any time, either using mp3 player ($\text{mp3p}$) on mp3 music or CD player on a CD, Peter can produce music (the choice $\oplus_2$ here is internal).

John’s resource (denoted as JR):

$$\Box [\Box \text{mp3p} \oplus_3 \Box \text{CDp}]$$

i.e. John can provide either an mp3 player or CD player after two days. $\oplus_3$ is an indeterminate possibility to Peter and will be revealed tomorrow.

We consider two strategies for Peter. If Peter does not let David borrow the CD burner, he can wait until tomorrow to learn what John will bring to the party and choose the music format to burn accordingly at that time. Otherwise, if he wants to let David borrow the CD burner, he can not delay burning the CD until tomorrow and so has to make a prediction on which player John will bring to the party, then decides the choice on the music format and burn the CD early (now). This corresponds to the second strategy. The question is then how to make such strategies available for agent Peter to explore.

We consider how Peter reasons along these strategies using TLL sequent calculus rules. Specifically, consider the following rules of sequent calculus that govern choices:

(set1) 

$$\frac{\Gamma, A \vdash \Delta \quad \Gamma, B \vdash \Delta}{\Gamma, A \oplus B \vdash \Delta} \quad \frac{\Gamma \vdash A, \Delta \quad \Gamma \vdash B, \Delta}{\Gamma \vdash A \& B, \Delta}$$

(set2) 

$$\frac{\Gamma, A \vdash \Delta \quad \Gamma, B \vdash \Delta}{\Gamma, A \& B \vdash \Delta} \quad \frac{\Gamma \vdash A, \Delta \quad \Gamma \vdash B, \Delta}{\Gamma \vdash A \oplus B, \Delta} \quad \frac{\Gamma \vdash A \oplus B, \Delta}{\Gamma \vdash A \& B, \Delta}$$

Formulas on the left hand side of $\vdash$ are considered as things that are available while formulas on the right hand side are considered as things that are in demand. Hence, $\&$ on
the left hand side of \( \vdash \) marks internal choice and \( \oplus \) marks indeterminate possibility and vice versa for when they are on the right hand side of \( \vdash \).

The first set of rules (set 1) reflects the strategy that agents prepare for all possible outcomes of indeterminate possibilities, whether they occur in formulas of what is available or what is in demand. The second set of rules (set 2) reflects agents’ decisions on internal choices at their associated times.

In the first strategy of Peter, he does not let David borrow the CD Burner. He can find a proof using standard sequent rules to achieve his goal of providing music at the party two days later (a proof of \( \bigcirc^2 m \)) as follows. Note that for space reasons, we combine a number of inference rule applications.

\[
\begin{align*}
\text{mp3} \vdash \text{mp3} & \hspace{1cm} \text{CDf} \vdash \text{CDf} \\
PR, PB \vdash \text{mp3} & \hspace{1cm} PR, PB \vdash \text{CDf} \\
PR, PB, \text{mp3p} \vdash \text{mp3} \oplus \text{mp3p} & \hspace{1cm} PR, PB, CDp \vdash \text{CDf} \oplus \text{CDp} \\
PP, PR, PB, \text{mp3p} \vdash \bigcirc m & \hspace{1cm} PP, PR, PB, CDp \vdash \bigcirc m \\
PP, PR, PB, \bigcirc m & \hspace{1cm} PP, PR, PB, JR \vdash \bigcirc^2 m
\end{align*}
\]

In this proof search, step two (labeled \( \oplus_3 \)) reflects that Peter prepares for all possibilities of the players (\( \bigcirc \text{mp3p} \oplus_3 \bigcirc \text{CDp} \)) the next day. The proof search succeeds because for whichever possibility occurs, Peter can use his CD burner to burn the CD in the correct format (CD or mp3).

However, in the second strategy where Peter makes his decision now, the search using standard TLL sequent calculus rules cannot find a proof of the goal \( \bigcirc^2 m \). At the search step labeled \( \oplus_3 \), it is necessary to deal with both cases of the music player, corresponding to two branches in the proof tree. In these branches, only the one in which Peter has chosen the correct music format succeeds. We present a proof search that reflects Peter’s choice of the music format as mp3 as an example. Note that in the proof search, the step labeled \( \&_1, \neg \) corresponds to Peter’s decision on the choice of music format. The search corresponds to the choice of CD format is quite similar.
It can be seen that even though Peter chooses the music format that will correctly match with the player, the search using standard sequent calculus rules still fails.

More generally, an important observation to make is that although (temporal) linear logic captures the notions of internal choice and indeterminate possibility, its sequent rules constrain decision making on them to certain strategies and to be done in isolation (subject only to local information).

Specifically, though the strategy of rules in set 1 is preparing all possible cases and hence safe, it requires extra and unnecessary resources and actions. Moreover, the strategy does not take into account the agents predictions of the outcomes of some indeterminate possibilities in the environment (which in our example is Peter’s prediction of the player) and whether agents are willing to take some risks by following their predictions. This also means that the notion of proof needs to be extended to allow for cases where it is not necessary for the proof to cover all possible outcomes of indeterminate possibilities but only their actual outcomes.

Furthermore, according to the second set of rules, the internal choices of agents are determined freely, without any guidance or constraints. Hence, such decisions about these choices may not be optimal. For example, if an agent decides an internal choice □A & □B to be □A via the third rule now and later realizes there are some goals that require □B then it misses the chance to obtain the necessary resource □A. Hence, it is important that information such as that about other goals is included in the agents’ consideration of decisions on internal choices. In addition, the sequent rules do not allow agents to explore the strategy of deciding choices in advance.

Hence, we investigate how we can use TLL not only to model the difference between internal choice and indeterminate possibility with respect to time, but also to capture dependencies among choices, constraints on how choices can be made and predictions and decisions of indeterminate possibilities. Such constraints may also reflect global consideration of other goals and other threads of interaction. We further consider strategies that can be used to deal with choices with respect to time, reflecting how cautious the agents are and whether the agents deal with them in advance. However, we will not discuss how agents can predict the environment outcomes correctly.

5.2 Modeling Decisions about Choices

In this section, we consider the modeling of choice decisions, how to express constraints on choice decisions, and their dependency on other choices’ decisions.
If we assume that the order of the operants is unchanged throughout the process, then the
decision on choices and indeterminate possibilities can be regarded as selecting the left-hand
side or the right-hand side of the connective. For simplicity, we shall refer to both internal
choices and indeterminate possibilities simply as choices unless it is important to mention
them distinctly.

Each choice is associated with a particular time point. Due to the inherent property of
TLL that formulas denoted at a specific time point exist only in that time point and becomes
invalid afterward, we assume that outcomes of choices must be revealed at their associated
times. To enable agents to decide internal choices in advance, we assume that agents can
make decisions early and keep those decisions unchanged until the decisions become effective.
Regarding indeterminate possibilities, we assume that their outcomes are determined by the
environment (or external factors) at their associated times, and only at these times do the
outcomes become known to the agents. For example, given an indeterminate possibility
\( \bigcirc^4(A \oplus B) \), after four time points, the environment determines that \( \bigcirc^4(A \oplus B) \) becomes
\( \bigcirc^4A \) or becomes \( \bigcirc^4B \).

Hence, for a given choice at the time \( \bigcirc^x, x > 0 \), there are three distinct states — not
determined, left chosen, and right chosen.

We use the notation \( \text{\&}_{x} \) or \( \oplus_{x} \) to record the outcomes on the choices \( \&_{x} \) and \( \oplus_{x} \) respectively.
The subscript indicates the ID of the connective. Base values for choice decisions can
be encoded by TLL propositions as \( L \) for choosing the left, and \( R \) for choosing the right.
For example, the base value for a decision on \( A \& B \) is \( L \) (denoted as \( \text{\&}_{= L} \) if \( A \) results
from deciding the choice \( A \& B \) (by agents or by the environment) and is \( R \) (\( \oplus_{= R} \) if \( B \)
results. Formally, we write \( \vdash \text{\&}_{1} \rightarrow \text{\&}_{0} \) or \( \vdash \text{\&}_{1} \rightarrow L \) to denote that the left sub-formula of \( \text{\&}_{1} \)
was selected. This notation distinctly captures the decision on choices out of their context
of resource or action relations.

Regarding internal choices, their decisions can be regarded as variables on which agents
can decide the assignment of values.

By modeling choices’ decisions explicitly, we can state constraints between them. For
example, if two choices, \( \&_{x} \) and \( \oplus_{y} \), need to be made consistently — either both right or both
left — then this can be stated as \( \text{\&}_{x} = \text{\&}_{y} \) or in the logic encoding, \( \text{\&}_{x} \vdash \text{\&}_{y} \), \( \text{\&}_{x} \vdash \text{\&}_{y} \).

More generally, we can state that a given internal choice \( \&_{x} \) (or \( \oplus_{x} \)) should depend on
a combination of other choices or some external constraints. We use \( \text{cond}L_{x} \) (respectively
\( \text{cond}R_{x} \)) to denote the condition that should hold for the left side (respectively right side) of
the internal choice to be taken.
Definition 2. Left Condition

Let time_\text{op}(\Gamma \&_x \Delta) \ (or \ time_\text{op}(\Gamma \oplus_x \Delta)) \ be \ a \ TLL \ formula, \ where \ time_\text{op} \ is \ a \ temporal \ operator \ or \ a \ combination \ of \ temporal \ operators. \ The \ “left \ condition”, \ denoted \ as \ condL_x, \ is \ defined \ as \ the \ necessary \ and \ sufficient \ conditions \ that \ the \ choice \ \&_x \ (or \ \oplus_x \ respectively) \ is \ determined \ such \ that \ time_\text{op}(\Gamma \&_x \Delta) \ (or \ time_\text{op}(\Gamma \oplus_x \Delta)) \ becomes \ time_\text{op}\Gamma.

Definition 3. Right Condition

Let time_\text{op}(\Gamma \&_x \Delta) \ (or \ time_\text{op}(\Gamma \oplus_x \Delta)) \ be \ a \ TLL \ formula, \ where \ time_\text{op} \ is \ a \ temporal \ operator \ or \ a \ combination \ of \ temporal \ operators. \ The \ “right \ condition”, \ denoted \ as \ condR_x, \ is \ defined \ as \ the \ necessary \ and \ sufficient \ conditions \ that \ the \ choice \ \&_x \ (or \ \oplus_x \ respectively) \ is \ determined \ such \ that \ time_\text{op}(\Gamma \&_x \Delta) \ (or \ time_\text{op}(\Gamma \oplus_x \Delta)) \ becomes \ time_\text{op}\Delta.

Clearly, satisfaction of condL_x means condR_x can not be satisfied and vice versa. In other words, they should always be mutually exclusive. These conditions completely determine the results of the choices' decisions. We encode these conditions as TLL sequents so that sub-conditions (sequents) can be found via proof search. As outcomes of indeterminate possibilities are not dependent on agents, for practical reasons, we do not specify constraints on indeterminate possibilities.

Consider our running example. Peter’s internal choice about the music format depends on which player John will bring to the party. Specifically, \(\&_1 = L \) (mp3 format) if and only if \(L \oplus_1 R \vdash R\) at the next day when Peter gets the information about the player from John. This dependency can be expressed as a constraint on the decision of \&_1. Specifically, the condition for the choice of \&_x being left chosen is defined as

\[ condL_1 \vdash \circ (\&_1 \rightarrow L) \]

It can be seen that internal choices with constraints express richer forms of internal choices, varying from free choices to indeterminate possibilities. Specifically, in the absence of any constraints, internal choices are truly free choices for agents. If the constraints contain some indeterminate possibilities then the outcomes of the respective internal choices are beyond the agents’ control. In this case, the internal choices behave similarly to indeterminate possibilities and the agents can only make predictions.

Hence, by modeling internal choices with constraints on their decisions, we naturally bridge the gap in the level of determinism between internal choices and indeterminate possibilities.
5.2.1 Representative Choice of a Subformula

Composite formulas, in general, contain sub-formulas that are interrelated. A sub-formula may be removed or retained after choices in the composite formula are decided. It is then important to state the conditions under which a sub-formula of interest is retained or removed. Based on such information, agents are then able to judge the significance of a compound formula w.r.t. retaining the sub-formula of interest and exercise various strategies in dealing with the compound formula.

Given the ability to model decisions about choices, especially representing constraints on decisions, we can capture whether a sub-formula is retained (or removed) in a composite formula by a choice. We can also express the conditions required to retain the sub-formula.

We start from an observation that given a formula \( \Gamma \) which contains a sub-formula \( A \), we can determine a sequence of decisions to be made to retain the sub-formula \( A \). For example, if

\[
\Gamma = B \otimes_1 \bigcirc^a (\bigcirc^b (A \oplus_2 C) \otimes_3 D)
\]

then in order to retain \( A \), we need to choose the right hand side of \( \otimes_1 \), then, \( a \) time units later, choose the left side of \( \otimes_3 \), and \( b \) time units after that, have the left side of \( \oplus_2 \) chosen by the environment (an indeterminate possibility).

This sequence of decisions about \( \Gamma \) to obtain \( A \) can be captured by the sequent:

\[
\vdash (\overset{\otimes_1}{\leftrightarrow} R) \otimes \bigcirc^a (\overset{\otimes_3}{\leftrightarrow} L) \otimes \bigcirc^{a+b} (\overset{\oplus_2}{\leftrightarrow} L)
\]

Observe that if we compress the sequence of decisions to retain \( A \) in \( \Gamma \) into a single choice, then the above sequent can be regarded as the determining condition for that choice.

In particular, in the above example, we can compress the decisions about \( \Gamma \) into a choice of the form \( \bigcirc^{a+b} A \otimes_\varphi \bigcirc^y 1 \) where the choice \( \otimes_\varphi \) is left chosen if \( \bigcirc^{a+b} A \) is retained in \( \Gamma \) and is right chosen otherwise. The condition sequent above then corresponds to \( \text{cond}_\varphi \) of the choice \( \otimes_\varphi \). Being mutually exclusive, \( \text{cond}_\varphi \) is captured as:

\[
\vdash (\overset{\bigcirc_1}{\leftrightarrow} L) \oplus \bigcirc^a (\overset{\bigcirc_3}{\leftrightarrow} R) \oplus \bigcirc^{a+b} (\overset{\bigcirc_2}{\leftrightarrow} R)
\]

Such a “compressed” choice is called a representative choice for \( A \). As the same choices are formulated with different notations on the left hand side and right hand side of \( \vdash \), we will define the notion of a representative choice in the context of program formulas and goal formulas respectively as follows.
Definition 4. Program Formula
Let \( \Gamma \vdash \Delta \) be a TLL sequent. A (TLL) formula \( A \) is said to be a program formula w.r.t. the given sequent iff \( A \) is an element of the multi-set \( \Gamma \).

Definition 5. Goal Formula
Let \( \Gamma \vdash \Delta \) be a TLL sequent. A (TLL) formula \( A \) is said to be a goal formula w.r.t. the given sequent iff \( A \) is an element of the multi-set \( \Delta \).

In other words, program formulas appear on the left hand side of \( \vdash \) while goal formulas appear on the right hand side. A “compressed” choice is then defined.

Definition 6. A representative choice \&_r, with respect to a formula \( A \) in a program formula (respectively goal formula) \( \Gamma \) is a choice \( \circ^x A \&_r \circ^y 1 \) (respectively \( \circ^x A \oplus_r \circ^y 1 \)) whose decision is \( L \) if \( \circ^x A \) is retained in \( \Gamma \) and is \( R \) otherwise (where \( \circ^y 1 \) is retained instead), where \( x \geq 0 \) is the time associated with \( A \) in \( \Gamma \) and \( y \geq 0 \) is the time point associated with 1.

Note that at the time of representing the choice \&_r (or \( \oplus_r \)), the value of \( y \) is not known. It will be known after all the decisions of internal choices and indeterminate possibilities in \( \Gamma \) are determined.

The representative choice of a sub-formula reflects the relationship with the compound formula that contains it. We will discuss strategies for agents to deal with such relationships in Section 5.4. In the meantime we discuss how standard sequent calculus rules might be extended to accommodate more strategies concerning choices.

5.3 Choice Calculus

Proof search using the standard sequent calculus rules reflect the strategy of preparing for all possible cases, and do not allow agents to take risks by choosing to prepare for only certain cases. Hence an extension to the sequent calculus rules is required to accommodate the possibility for agents to follow a risky-taking strategy.

In particular, we need to provide the proof steps in which agents can make predictions on the outcomes of indeterminate possibilities and follow only the search paths corresponding to the predicted ones. In other words, we need to provide inference rules for dealing with indeterminate possibilities such that the agents can decide on a particular branch to follow in proof search rather than preparing for all branches. We also need to keep track of the predictions that agents make. Sequent rules of this form are below. Note that \( \vdash_{cc} \) means \( \vdash \)
in the choice calculus and a prediction of the outcome of an indeterminate possibility $\oplus_n$ is expressed in square brackets.

\[
\begin{align*}
\Gamma \vdash_{cc} F, \Delta & \quad \vdash_{\oplus_n} L \\
\Gamma \vdash_{cc} F \oplus_n G, \Delta & \quad \vdash_{\oplus_n} R
\end{align*}
\]

The predictions are supposed to be evaluated independently from the main formulas in the inference rules and at the time when the environment reveals the outcomes.

A prediction can be thought as an assumption that the search technique relies on and needs to be checked against the actual outcomes. If the predictions are correct, as in the standard case, a successful search will generate a proof. If the assumptions are not correct, then even if the search is successful, no proof is generated. Hence the existence of a proof depends on the agent’s predictions being correct. We will discuss the relationship between proofs using the extended inference rules and proofs using only standard sequent calculus rules in Section 5.3.1.

Moreover, we allow agents to decide upon an indeterminate possibility beforehand, which is not possible using only the standard sequent rules of TLL. What is needed then are inference rules that permit agents to follow only the branches that correspond to their predictions on the indeterminate possibilities. Rules of this form are below. As above, predictions are in square brackets.

\[
\begin{align*}
\Gamma \vdash_{cc} O^n F, \Delta & \quad \vdash_{\oplus_n} O^n (F \oplus_n G), \Delta \\
\Gamma, O^n F \vdash_{cc} \Delta & \quad \vdash_{\oplus_n} O^n (F \oplus_n G), \Delta
\end{align*}
\]

Internal choices can be decided by the owner agent at the time associated with the choices, subject to any constraints ($\text{cond}L_n$ or $\text{cond}R_n$) imposed on them. Accordingly, we derive new sequent rules for internal choices to reflect that such constraints need to be followed by attaching the corresponding conditions to the proof search steps. The new sequent rules are:
Choice Calculus

$$\begin{align*}
\Gamma, F \vdash_{cc} & \Delta \quad [\text{cond}_{L_n}] \\
\Gamma, F \& n G \vdash_{cc} & \Delta \\
\Gamma \vdash_{cc} & F; \Delta \quad [\text{cond}_{L_n}] \\
\Gamma \vdash_{cc} & F \oplus n G, \Delta \\
\Gamma, G \vdash_{cc} & \Delta \quad [\text{cond}_{R_n}] \\
\Gamma, F \& n G \vdash_{cc} & \Delta \\
\Gamma \vdash_{cc} & F \oplus n G, \Delta \\
\end{align*}$$

where \( \text{cond}_{L_n} \) (respectively \( \text{cond}_{R_n} \)) are conditions imposed on the internal choice \( n \) for the choice to be left (respectively right) chosen. These conditions may be absent. In their absence, the internal choices are truly free choices and the standard inference rules are used.

Moreover, if the agent is to decide the choice in advance, it can use the choice’s outcome earlier in the search. Similar to the rules that allow agents to deal with indeterminate possibilities ahead of time, the rules should permit the agents to follow a particular choice while keeping track of the associated conditions. They are of the form:

$$\begin{align*}
\Gamma; \circ^n F \vdash_{cc} & \Delta \quad [\text{cond}_{L_n}] \\
\Gamma; \circ^n (F \& n G) \vdash_{cc} & \Delta \\
\Gamma \vdash_{cc} & \circ^n F; \Delta \quad [\text{cond}_{L_n}] \\
\Gamma \vdash_{cc} & \circ^n (F \oplus n G), \Delta \\
\Gamma; \circ^n G \vdash_{cc} & \circ^n \Delta \quad [\text{cond}_{R_n}] \\
\Gamma; \circ^n (F \& n G) \vdash_{cc} & \circ^n \Delta \\
\Gamma \vdash_{cc} & \circ^n F \oplus n G, \Delta \\
\end{align*}$$

These above new sequent rules, together with standard TLL sequent rules, form the choice calculus. The new sequent rules are summarized in Table 5.1.

Let us return to our running example. Recall that if Peter is to let David borrow the CD burner now, then he needs to make a prediction about the player that John will bring, and based on this prediction, decide on the indeterminate possibility early. For instance, Peter predicts that John will provide an mp3 player (i.e. \( L \oplus_3 R \vdash L \)). Using the choice calculus, this is captured by the following inference:

$$\begin{align*}
\Gamma; \circ^2 \text{mp3p} \vdash_{cc} \circ^2 m \quad [\vdash \circ^{(\text{mp3p} \oplus_3 \text{CDp})} \vdash_{cc} \circ^2 m] \\
\end{align*}$$

where \( \Gamma \) is some formula in the proof.

Based on this prediction, Peter decides on the choice of music format \( \&_1 \) (mp3 format) now and burns the blank CD accordingly. As mentioned earlier, the imposed condition for the choice \( \&_1 \) is \( \text{cond}_{L_1} \vdash \circ^{(\text{mp3p} \oplus_3 \text{CDp})} \vdash_{cc} \circ^2 m \). Such a decision with a constraint is reflected in the following proof step in the choice calculus.
\[
\Gamma, \Box mp3 \vdash \Delta \quad \vdash \bigcirc (\Box^3 \rightarrow L) \\
\Gamma, \Box mp3 \&_1 \Box CDf \vdash \Delta
\]

where \( \Gamma \) and \( \Delta \) are some formulas in the proof.

By taking the risk in following his prediction, Peter then successfully obtains a proof of \( \bigcirc^2 m \) (given below). If his prediction is correct, which means that \( L \oplus_3 R \vdash L \) is provable, then a successful plan is obtained to achieve his goal.

We have the following proof of \( \bigcirc^2 m \) in the choice calculus where some inferences combine a number of rule applications.

\[
\begin{align*}
\Gamma, \Box mp3 &\vdash \Delta \quad \vdash \bigcirc (\Box^3 \rightarrow L) \\
\Gamma, \Box mp3 \&_1 \Box CDf &\vdash \Delta
\end{align*}
\]

In this example we begin (bottom-most inference) by making an “in-advance” decision for the indeterminate possibility \( \oplus_3 \). Specifically, we predict that John will provide an MP3 player. We then use standard the sequent rule for internal choice to decide on the format MP3. When the time comes to make a decision for \( \oplus_3 \) we can select to use the MP3 player to produce music. As the condition \( \text{cond}L_1 = \vdash \bigcirc (\Box^3 \rightarrow L) \) is the same as the condition of the prediction on \( \oplus_3 \), we omit it for readability.

As can be seen from the example, internal choices and indeterminate possibilities are properly modeled with respect to time. Moreover, several strategies are enabled for Peter due to the use of the choice calculus. If Peter is to take a safe approach, he should delay deciding the music format until tomorrow and ignores David’s request. If Peter is willing to take risks, he can predict the indeterminate possibility of which player John will bring to the party and act accordingly. Peter can also decide the choice on music early so as to lend David the CD burner.

To illustrate the use of the choice calculus in handling choices modeled with constraints on their decisions, we give an example by considering the following formulas.

\[
(\bigcirc^n A \&_a 1) \oplus (\bigcirc^n B \&_b 1) \oplus (C \&_c 1)
\]
where \( \&_a, \&_b, \&_c \) are internal choices.

We consider some constraints for the internal choices which are specified out the outcomes of other choices, namely \( \&_1, \oplus_2 \) and \( \oplus_3 \) as follows.

In particular, we define the conditions for the choice \( \&_a \) being left chosen as

\[
\text{cond}_{L_a} = \vdash \bigcirc^n(\overset{\&_1}{\rightarrow} R) \otimes (\overset{\oplus_2}{\rightarrow} L)
\]

The conditions for the choice \( \&_a \) being right chosen are defined in a mutually exclusive manner as

\[
\text{cond}_{R_a} = \vdash \bigcirc^n(\overset{\&_1}{\rightarrow} L) \oplus (\overset{\oplus_2}{\rightarrow} R)
\]

Similarly for \( \&_b \)

\[
\text{cond}_{L_b} = \vdash \bigcirc^n(\overset{\&_1}{\rightarrow} L) \otimes (\overset{\oplus_2}{\rightarrow} L), \quad \text{cond}_{R_b} = \vdash \bigcirc^n(\overset{\&_1}{\rightarrow} R) \oplus (\overset{\oplus_2}{\rightarrow} R)
\]

and \( \&_c \)

\[
\text{cond}_{L_c} = \vdash \overset{\oplus_2}{\rightarrow} R \quad \text{cond}_{R_c} = \vdash \overset{\oplus_2}{\rightarrow} L
\]

To illustrate proof search in the choice calculus, we consider another formula that also has the choice \( \oplus_3 \cdot \bigcirc^n B \otimes (\bigcirc^n A \oplus_3 C) \) - and check if

\[
\bigcirc^n B \&_b 1, \bigcirc^n A \&_a 1, C \&_c 1 \vdash \bigcirc^n B \otimes (\bigcirc^n A \oplus_3 C)
\]

is the case.

One approach to proof search techniques in the choice calculus is to delay evaluation of all conditions until the search is complete. Such an approach is illustrated as below.

\[
\begin{array}{c}
B \vdash B \quad \text{[cond}_{L_b}] \\
\bigcirc^n B \vdash \bigcirc^n B \quad \text{[cond}_{L_b}] \\
\bigcirc^n B \&_b 1 \vdash \bigcirc^n B \\
\bigcirc^n B \&_b 1 \bigcirc^n B \quad \text{[cond}_{L_b}] \\
\end{array}
\]

\[
\begin{array}{c}
A \vdash A \quad \text{[cond}_{L_a}, \text{cond}_{R_c}] \\
\bigcirc^n A \vdash \bigcirc^n A \\
\bigcirc^n A, 1 \vdash \bigcirc^n A \quad \text{[cond}_{L_a}, \text{cond}_{R_c}] \\
\bigcirc^n A, 1, C \&_c 1 \vdash \bigcirc^n A \quad \text{[cond}_{L_a}] \\
\bigcirc^n A, 1, C \&_c 1 \vdash \bigcirc^n A \oplus_3 C \\
\bigcirc^n A, 1, C \&_c 1 \vdash \bigcirc^n B \otimes (\bigcirc^n A \oplus_3 C) \\
\end{array}
\]

The conditions required on the proof are a conjunction of all the conditions of its branches. It is expressed in TLL as \( \vdash \text{cond}_{L_b} \otimes \text{cond}_{L_c} \otimes \text{cond}_{L_a} \). A proof of the conditions then requires:
In order to satisfy all the conditions, the following decisions on choices need to occur:

\[ \begin{align*}
\delta_1 &\rightarrow L, \delta_2 = L, \delta_2 = R, \delta_1 = L, \\
\end{align*} \]

in which there is an inconsistency as \( \delta_1 \) must have both values \( L \) and \( R \). Hence, the conditions cannot all be satisfied and so there is no proof.

The proof search can backtrack at the proof step labeled \( \oplus_3 \) to take the other path as follows.

The conditions required on the proof are \( \vdash \text{cond}_{L_b} \otimes \text{cond}_{L_c} \otimes \text{cond}_{R_a} \). A proof of the conditions is below

We then have the following requirements:

\[ \begin{align*}
\delta_1 &\rightarrow L, \delta_2 = L, \delta_2 = R, \delta_1 = L, \\
\end{align*} \]

from which we get that \( \delta_2 \) must have both values \( L \) and \( R \), which is inconsistent. The proof search of the conditions then backtracks at the step labeled \( \oplus \) to take the other path as follows.
The corresponding requirements are:

\[
\begin{align*}
\text{cond}_L b & = L, \\
\text{cond}_L c & = L, \\
\text{cond}_\oplus 2 & = R, \\
\text{cond}_\oplus 2 & = R,
\end{align*}
\]

from which we get that \text{cond}_\oplus 2 must have both values \(L\) and \(R\), which is inconsistent. Hence, the conditions cannot all be satisfied and we cannot obtain any proof of \(\bigcirc^n B \otimes (\bigcirc^n A \oplus_C C)\) whose conditions can be satisfied.

Another possible proof search technique for the choice calculus is to evaluate conditions as they occur in proof steps. As newly formed conditions are satisfied, the proof search continues further. Otherwise, an unfulfilled condition will stop the search and causes it to backtrack if possible. The example below illustrates the technique.

Assume that the search reaches this point

\[
\begin{align*}
B \vdash B & \quad [\text{cond}_L b] \\
\bigcirc^n B & \vdash \bigcirc^n B \ [\text{cond}_L b] \\
\bigcirc^n B & \&_b 1 \vdash \bigcirc^n B \ &_{\text{cond}_L b} \\
\bigcirc^n B & \&_b 1, \bigcirc^n A \&_a 1, C \&_c 1 \vdash \bigcirc^n A \oplus_C C \\
\bigcirc^n B \otimes (\bigcirc^n A \oplus_C C) & \vdash \text{cond}_L b
\end{align*}
\]

The condition that needs to be satisfied is \text{cond}_L b. A proof requires the decision on the choice \text{cond}_1 to be \(L\) and also the decision on the choice \text{cond}_2 to be \(L\) as follows:

\[
\begin{align*}
\text{cond}_1 & = L, \\
\text{cond}_2 & = L
\end{align*}
\]

The proof search goes further and reaches another condition \text{cond}_L c
The proof of \( \text{cond}_L \) requires

\[
\frac{\text{\( \text{cond}_L \)}}{\vdash R}
\]

which means the decision for the choice \( \oplus_2 \) is \( R \) (\( \text{\( \rightarrow \)} \) = \( R \)). This is inconsistent with the previous condition \( \text{cond}_L \), in which the decision on the choice \( \oplus_2 \) is \( L \). As a result, the proof search stops and backtracks to take another path.

The choice calculus allows agents to use various strategies for indeterminate possibilities and internal choices. These strategies make it more flexible to deal with changes and handle exceptions with global awareness and dependencies among choices. Moreover, because each choice has one decision value (being left or right chosen), in order to keep decisions on choices consistent, there needs to be in place a mechanism that keeps track of all the decisions on choices and makes them consistent. Developing such mechanism is a straightforward implementation task and is not discussed in the thesis. In the next section, we show that proofs that also use the additional rules are, in a sense, equivalent to proofs in the original TLL sequent calculus.

### 5.3.1 Soundness and Completeness of the Choice Calculus

The intuition behind the soundness and completeness properties of proofs using these additional rules with respect to proofs which only use original TLL sequent calculus is that eventually indeterminate possibilities will be determined. Hence, if the agents have made the correct predictions and followed them then they have successfully dealt with these indeterminate possibilities.

The soundness and completeness properties are then evaluated and proved in this context. In particular, we introduce the concept of a revealed proof, which has all the internal choices and indeterminate possibilities replaced by their actual outcomes. Proofs under the choice calculus are then examined in relation to their corresponding revealed proofs. They are sound if all the assumptions they rely on turn out to be correct. If the assumptions turn out to be
unfounded, then the proofs under the choice calculus are not valid.

We firstly define the concept of a revealed proof.

**Definition 7.** Let $\Gamma \vdash \Delta$ denote a sequent in TLL that does not contain any choice in the formula. A revealed proof of $\Gamma \vdash \Delta$ is a proof under standard sequent calculus rules which does not contain any inference rules over choices.

A formula appearing in a revealed proof is called a revealed formula.

By definition, a revealed proof is also a valid proof under standard TLL sequent calculus rules.

A proof under the choice calculus is then different from its corresponding revealed proof essentially in that occurrences of choices have not yet revealed their outcomes. The process of revealing choices that turns a proof under the choice calculus into a revealed proof is defined as follows.

**Definition 8.** Let $P$ be a proof of a sequent $\Gamma \vdash_{cc} \Delta$ in the choice calculus. A transformation process $R_{choice}$ applied to $P$ replaces any occurrences of choices in all sequents in $P$ by their actual outcomes.

Note that, regarding additional inference rules in the choice calculus, by definition, the conditions attached to them hold if and only if the premise sequents correspond to the actual outcomes of the choices. As the transformation $R_{choice}$ turns all the choices in conclusion sequents to their actual outcomes, $R_{choice}$ makes the premise sequents the same as the conclusion sequents after the transformation.

Such replacements by $R_{choice}$ manifest in changes to inference rules on choices as shown in the examples below. Indeed, if all the conditions of inference rules on choices in $P$ are correct, these inference rules are turned into identity rules, which can be safely replaced by one sequent. We list a few cases and their corresponding identities rules are listed immediately on their right hand side. Listing the rest is straightforward and repetitive, and hence is omitted here. Let $\Gamma'$, $F'$, $G'$ and $\Delta'$ denote the outcomes of replacing all choices in $\Gamma$, $F$, $G$ and $\Delta$ by their actual outcomes respectively.

\[
\frac{\Gamma \vdash_{cc} F, \Delta \quad [\rightarrow_n L]}{\Gamma \vdash_{cc} F \rightarrow_n G, \Delta} \quad \frac{\Gamma' \vdash F', \Delta'}{\Gamma' \vdash F', \Delta'}
\]

\[
\frac{\Gamma \vdash_{cc} F, \Delta \quad [\text{cond}\Gamma_n]}{\Gamma \vdash_{cc} F \oplus_{n} G, \Delta} \quad \frac{\Gamma' \vdash F', \Delta'}{\Gamma' \vdash F', \Delta'}
\]
\[
\frac{\Gamma, F \vdash cc \Delta \quad \vdash n \rightarrow L}{\Gamma, F \oplus_n G \vdash cc \Delta}
\]
\[
\frac{\Gamma, F \oplus_n G \vdash cc \Delta}{\Gamma', F' \vdash \Delta'}
\]
\[
\frac{\Gamma, G \vdash cc \Delta \quad \text{[cond} R_n]}{\Gamma, F \&_n G \vdash cc \Delta}
\]
\[
\frac{\Gamma, F \&_n G \vdash cc \Delta}{\Gamma', G' \vdash \Delta'}
\]
\[
\frac{\Gamma, \bigcirc^n F \vdash cc \Delta \quad \text{[cond} L_n]}{\Gamma, \bigcirc^n (F \&_n G) \vdash cc \Delta}
\]
\[
\frac{\Gamma, \bigcirc^n (F \&_n G) \vdash cc \Delta}{\Gamma', \bigcirc^n F' \vdash \Delta'}
\]
\[
\frac{\Gamma \vdash cc \bigcirc^n F, \Delta \quad \text{[cond} L_n]}{\Gamma \vdash cc \bigcirc^n (F \oplus_n G), \Delta}
\]
\[
\frac{\Gamma \vdash cc \bigcirc^n (F \oplus_n G), \Delta}{\Gamma', \bigcirc^n F' \Delta'}
\]
\[
\frac{\Gamma \vdash cc F, \Delta \quad \Gamma \vdash cc G, \Delta}{\Gamma \vdash cc (F \&_n G), \Delta}
\]
\[
\frac{\Gamma \vdash cc (F \&_n G), \Delta}{\Gamma' \vdash F', \Delta'} \quad \text{(F is the actual outcome)}
\]
\[
\frac{\Gamma, F \vdash cc \Delta \quad \Gamma, G \vdash cc \Delta}{\Gamma, (F \oplus_n G) \vdash cc \Delta}
\]
\[
\frac{\Gamma, (F \oplus_n G) \vdash cc \Delta}{\Gamma' \vdash F', \Delta'} \quad \text{(F is the actual outcome)}
\]

It can be seen that as a result of such replacements, all the inference rules on choices collapse to identity rules.

Consider the following proof steps in the choice calculus.

\[
\frac{C \vdash C \quad \vdash n \rightarrow R}{B \vdash B \quad C \&_2 D \vdash C \quad \text{[\text{\(\oplus_1\)} \rightarrow R]} \quad \text{\(\&_2\)}}
\]
\[
\frac{B, C \&_2 D \vdash B \quad \text{[\text{\(\oplus_1\)} \rightarrow R]} \quad \text{\(\&\)}}}{A \&_1 B, C \&_2 D \vdash B \& C \quad \text{\(\oplus_1\)}}
\]

Applying the transformation \(\text{Re}_{\text{choice}}\) on the proof steps above will result in the following revealed proof (assuming that the conditions \[\vdash n \rightarrow R\] are correct, which means that the indeterminate possibility \(A \&_1 B\) turns out to be \(B\)).

\[
\frac{B \vdash B \quad C \vdash C}{B, C \vdash B \& C \quad \text{\(\&\)}}\]
\[
\frac{B \vdash B \quad C \vdash C}{B, C \vdash B \& C}
\]
which essentially is

\[
\frac{B \vdash B \quad C \vdash C}{B, C \vdash B \otimes C} \otimes R
\]

We then have the following result regarding the outcome of the transformation \( R_{\text{choice}} \). Note that for simplicity of discussion, regarding a sequent \( \Gamma \vdash_{cc} \Delta \) we make use of the equivalent form \( \Gamma, \Delta^\perp \vdash_{cc} \).

**Theorem 5.3.1.** Let \( P \) be a proof of a sequent \( \Gamma_n \vdash_{cc} \) under the choice calculus where \( \Gamma_n \) is a formula of TLL. Let \( \text{rep}(\Gamma_n) \) be the outcome of replacing all the choices in \( \Gamma_n \) by their actual outcomes. Let \( \text{rep}(P) \) be the result of applying the transformation \( R_{\text{choice}} \) to \( P \).

We have that if all the conditions associated with additional inference rules over choices in \( P \) (if exist) are correct, \( \text{rep}(P) \) is a proof of \( \text{rep}(\Gamma_n) \vdash \) in the standard TLL sequent calculus.

**Proof:**

The theorem is proved by induction on the proof length of \( P \).

The base case is \( P \) of length 1, which contains an axiom rule. \( \text{rep}(P) \) is also the axiom rule and the theorem holds.

We assume the theorem holds for all \( P \) of length up to and including \( n \) proof steps and try to prove the case for \( (n+1) \) proof steps.

Let \( P_m \) denote a proof of length \( m \) proof steps. Let \( P'_m \) be the result of applying \( R_{\text{choice}} \) on \( P_m \). Let \( S_m = \Gamma_m, M_m \vdash_{cc} \) be the last sequent of \( P_m \) after \( m \) proof steps. Let \( S'_m = \Gamma'_m, M'_m \vdash_{cc} \) be the result of replacing all choices in \( S_m \) by their actual outcomes.

Hypothesis: \( P'_m \) is a proof of \( S'_m = \Gamma'_m, M'_m \vdash \) under standard TLL sequent calculus for all values of \( m \) that \( 1 < m \leq n \).

Let \( P_{n+1} \) denote the proof \( P \) of length \( n+1 \) proof steps. Let \( P'_{n+1} \) be the result of applying \( R_{\text{choice}} \) on \( P_{n+1} \).

Let \( n' + 1 \) denote the number of proof steps in \( P'_{n+1} \). Let \( R_{n+1} \) and \( R'_{n+1} \) denote the inference rules of \( P_{n+1} \) and \( P'_{n+1} \) respectively at step \( n + 1^{th} \) and \( n' + 1^{th} \) respectively.

Let \( S_{n+1} = \Gamma_{n+1}, M_{n+1} \vdash_{cc} \) be the last sequent of \( P \) after \( n + 1 \) proof steps, where \( M_{n+1} \) and \( \Gamma_{n+1} \) are the main formulas and context formulas with respect to \( R_{n+1} \). Let \( S'_{n+1} = \Gamma'_{n+1}, M'_{n+1} \vdash_{cc} \) be the result of replacing all choices in \( \Gamma_{n+1}, M_{n+1} \vdash_{cc} \) by their actual outcomes.

We need to establish that \( P'_{n+1} \) is a proof of \( \Gamma'_{n+1}, M'_{n+1} \vdash \) in the standard TLL sequent calculus.
We will do that by examining the premises and conclusion sequents with respect to various cases of the last inference rule in $P'_{n+1}$.

**Firstly**, because the transformation $Re_{\text{choice}}$ maps a single conclusion sequent to another single conclusion sequent, the last sequent in $P'_{n+1}$ is also a single conclusion sequent.

**Secondly**, we consider how the theorem holds with respect to the premises sequents of the last inference rule in $P_{n+1}$.

Note that for all inference rules, their premises contain at most two sequents.

Let $\Gamma_a, M_a \vdash \text{cc}$ and $\Gamma_b, M_b \vdash \text{cc}$ be two premises sequents of $R_{n+1}$, where $M_a$, $M_b$ denote the main formulas and $\Gamma_a$ and $\Gamma_b$ denote the context formulas with respect to the inference rule $R_{n+1}$ (note that $\Gamma_b, M_b \vdash \text{cc}$ may not exist). $R_{n+1}$ is

\[
\begin{align*}
\vdots & \quad \vdots \\
\Gamma_a, M_a & \vdash \text{cc} \quad \Gamma_b, M_b \vdash \text{cc} \\
\hline \\
\Gamma_{n+1}, M_{n+1} & \vdash \text{cc}
\end{align*}
\]

Under $Re_{\text{choice}}$, $R_{n+1}$ is turned into $R'_{n+1}$ where $R'_{n+1}$ is

\[
\begin{align*}
\vdots & \quad \vdots \\
\Gamma'_a, M'_a & \vdash \quad \Gamma'_b, M'_b \vdash \\
\hline \\
\Gamma'_{n+1}, M'_{n+1} & \vdash
\end{align*}
\]

where we do not know if $R'_{n+1}$ is valid as yet.

Given that $P_{n+1}$ is a proof, each sequent of the premises of the inference rule $R_{n+1}$ is also a last sequent of a sub-proof in $P_{n+1}$ of length less than or equal to $n$.

Denote $P_a$ and $P_b$ be the two sub-proofs in $P_{n+1}$ that have $\Gamma_a, M_a \vdash \text{cc}$ and $\Gamma_b, M_b \vdash \text{cc}$ as end sequents respectively ($P_b$ will not exist if $\Gamma_b, M_b \vdash \text{cc}$ does not exist).

Let $P'_a$ and $P'_b$ be the outcomes of the transformation $Re_{\text{choice}}$ on $P_a$ and $P_b$ respectively. Let $\Gamma'_a$, $M'_a$, $\Gamma'_b$, and $M'_b$ be the outcomes of replacing all occurrences of choices in $\Gamma_a, M_a, \Gamma_b$ and $M_b$ respectively.

Because $P_a$ and $P_b$ are proofs of length less than or equal to $n$, by the hypothesis, $P'_a$ and $P'_b$ are also proofs of the sequents $\Gamma'_a, M'_a \vdash \text{cc}$ and $\Gamma'_b, M'_b \vdash$ respectively in the standard TLL sequent calculus.
In other words, under $Re_{\text{choice}}$ on $P_{n+1}$, $P'_a$ and $P'_b$ result and are proofs of their last sequents $\Gamma'_a, M'_a \vdash_{cc}$ and $\Gamma'_b, M'_b \vdash_{cc}$. Hence, $\Gamma'_a, M'_a \vdash_{cc}$ and $\Gamma'_b, M'_b \vdash_{cc}$ are also the two premises sequents of $R'_{n+1}$ ($\Gamma'_b, M'_b \vdash_{cc}$ will not exist if $\Gamma_b, M_b \vdash_{cc}$ does not exist).

**Thirdly**, given that the theorem holds for the premises sequents of $R'_{n+1}$, we will establish that from these premises, $\Gamma'_{n+1}, M'_{n+1} \vdash$ is the conclusion sequent, where $\Gamma'_{n+1}, M'_{n+1} \vdash$ results from replacing all choices in $\Gamma_{n+1}, M_{n+1} \vdash_{cc}$ by their actual outcomes by examining all possible cases of $R_{n+1}$.

The inference rule $R_{n+1}$ can be either a choice rule (i.e. one from the choice calculus only) or a non-choice rule (i.e. one in standard TLL but not in the choice calculus).

**Case 1**: $R_{n+1}$ is an inference rule on choice.

There are two cases of $R_{n+1}$ in which the premises contain two sequents (note that $\Gamma_a = \Gamma_b$):

$$
\frac{\Gamma_a, M_a \vdash_{cc} \Gamma_a, M_b \vdash_{cc}}{\Gamma_a, M_a \oplus M_b \vdash_{cc}} \oplus L
$$

or

$$
\frac{\Gamma_a \vdash_{cc} M_a \Gamma_a \vdash_{cc} M_b}{\Gamma_a \vdash_{cc} M_a \& M_b} \& R
$$

In these cases, $\Gamma_{n+1} = \Gamma_a = \Gamma_b$, and hence the outcomes of replacing all choices in them by actual choice outcomes are the same, and so $\Gamma'_{n+1} = \Gamma'_a = \Gamma'_b$.

By applying $R_{n+1}$, another choice is introduced in $M_{n+1}$, $M_{n+1} = M_a \& M_b$ or $M_{n+1} = M_a \oplus M_b$. Assuming that the outcome of this choice is $M_a$, then replacing all the choices in $M_{n+1}$ by their outcomes results in $M_a$. Hence $M'_{n+1} = M_a$.

Furthermore, consider the transformation $Re_{\text{choice}}$ on $R_{n+1}$. $Re_{\text{choice}}$ turns the newly introduced choice in $M_{n+1}$ to its outcome, which means also that $M_{n+1}$ becomes $M_a$. After performing $Re_{\text{choice}}$, $R'_{n+1}$ is

$$
\frac{\Gamma'_a, M'_a \vdash}{\Gamma'_a, M'_a \vdash}
$$

Given that $M'_{n+1} = M'_a$ as above, we have the following proof in the standard TLL sequent calculus.

$$
\frac{\Gamma'_a, M'_a \vdash}{\Gamma'_a, M'_{n+1} \vdash}
$$
which means that the theorem holds for these two cases of $R_{n+1}$.

Consider all other cases of $R_{n+1}$ of inference rules over choices, the premises of $R_{n+1}$ then have only one sequent, denoted as $\Gamma_a, M_a \vdash cc$. As the only choice introduced to $M_a$ results in $M_{n+1}$, $\Gamma_a$ is unchanged. Hence, $\Gamma_{n+1} = \Gamma_a$, and the result of replacing all choices by their outcomes is the same, and so $\Gamma_{n+1}' = \Gamma_a'$.

By applying the inference rule $R_{n+1}$, $M_a$ becomes $M_{n+1}$. Now the only change in $M_{n+1}$ from $M_a$ is the newly introduced choice but it becomes its outcome under $R_{\text{choice}}$. Given all the conditions associated with additional inference rules over choices in $P$ are correct, the outcome of the newly introduced choice in $M_{n+1}$ is $M_a$. Consequently, the outcomes of replacing all choices in $M_a$ and $M_{n+1}$ are the same, and so $M_a' = M_{n+1}'$.

We then have a proof under standard TLL sequent calculus as follows

\[
\frac{\Gamma_a', M_a' \vdash}{\Gamma_a', M_{n+1}' \vdash}
\]

which means that the theorem holds for $R_{n+1}$.

**Case 2:** $R_{n+1}$ is a non-choice inference rule.

Among all non-choice inference rules, $R_{n+1}$ can have premises of two sequents or one sequent

\[
\frac{\Gamma_a, M_a \vdash cc \quad \Gamma_b, M_b \vdash cc}{\Gamma_{n+1}, M_{n+1} \vdash cc}
\]

or

\[
\frac{\Gamma_a, M_a \vdash cc}{\Gamma_{n+1}, M_{n+1} \vdash cc}
\]

$\Gamma_{n+1}$ can be either the same as $\Gamma_a$ (and $\Gamma_b$ if it exists) or combining them ($\Gamma_{n+1} = \Gamma_a, \Gamma_b$). Correspondingly, $\Gamma_{n+1}'$ can be either the same as $\Gamma_a'$ and $\Gamma_b'$ or combining them ($\Gamma_{n+1}' = \Gamma_a', \Gamma_b'$).

Observe that in a proof $P_{n+1}$ under the choice calculus, after a choice is introduced, like $A \& B$, its components ($A$ or $B$) will never be used individually and the choice will be used as a single and indivisible unit in further inference rules. In fact, if we treat all the choices in all the sequents in $R_{n+1}$ as black boxes (i.e. without knowing the content inside), then the non-choice inference $R_{n+1}$ still holds.

Consequently, the same non-choice inference rule as of $R_{n+1}$ in the standard TLL sequent
calculus can still be applied where, in the black boxes, choices are replaced by their actual outcomes. Hence, we have a valid inference rule as:

$$\frac{\Gamma_a', M_a' \vdash \Gamma_b', M_b' \vdash}{\Gamma_{n+1}', M_{n+1}' \vdash}$$

or

$$\frac{\Gamma_a', M_a' \vdash}{\Gamma_{n+1}', M_{n+1}' \vdash}$$

Hence, for all the cases of $R_{n+1}$, we have further established that the sequent $\Gamma_{n+1}', M_{n+1}' \vdash$ is the conclusion in the last inference step $R_{n+1}'$ of $P_{n+1}'$.

Given also that the premises of $R_{n+1}'$ are conclusion sequents of proofs that are sub-parts of $P_{n+1}'$ as established above, $P_{n+1}'$ as a whole is a proof of $\Gamma_{n+1}', M_{n+1}' \vdash$. In other words, we have shown that the theorem holds for proofs of length $n + 1$.

By induction on the length of proofs, we can conclude that the theorem holds. $\square$

As a result of this theorem, it is straightforward to obtain the soundness property as follows.

**Theorem 5.3.2. (Soundness)** Let $P$ be a proof of $\Gamma \vdash_{cc} \Delta$ in the choice calculus where $\Gamma$ and $\Delta$ are formulas of TLL. Let $\text{rep}(\Gamma)$ and $\text{rep}(\Delta)$ be the outcomes of replacing all the choices in $\Gamma$ and $\Delta$ respectively by their outcomes. Let $\text{rep}(P)$ be the result of applying the transformation $\text{Re}_{\text{choice}}$ to $P$.

If all the conditions associated with the additional inference rules on choices in $P$ are correct,

$$\text{rep}(\Gamma) \vdash \text{rep}(\Delta)$$

is also provable in standard TLL sequent calculus by the proof $\text{rep}(P)$.

**Proof:** Follows trivially from Theorem 5.3.1 $\square$

The completeness theorem is as follows.

**Theorem 5.3.3. (Completeness)** Let $\Gamma$, $\Delta$ be formulas of TLL. If $\Gamma \vdash \Delta$ is provable in the standard TLL sequent calculus then $\Gamma \vdash_{cc} \Delta$ is provable in the choice calculus.

**Proof:** As the choice calculus also contains standard TLL sequent calculus rules, this is trivial. $\square$
5.4 Dealing with Choices and Changes

The choice calculus lets agents exercise various strategies on choices, such as preparing for all possible cases (being cautious), or assuming some particular outcomes by making predictions (being bold). The rationale for the bold options is to save effort by cutting down the number of possibilities to deal with. Similarly, deciding choices beforehand has the advantage of reducing the number of future options/possibilities that one has to prepare for. This applies particularly during interactions, in which resources and actions are offered to other agents.

As agents decide choices during interaction, there are two extreme cases.

The first is that all the choices and indeterminate possibilities (now and future) are decided. As a result, the resource requirements to fulfill commitments are at minimum and agents do not have to reserve more resources than what will be consumed. The requirements are also straightforward and agents can use resources to handle them with certainty.

The other extreme is that all decisions for choices and indeterminate possibilities are left open. In this case, in order to fulfill goals and commitments, agents prepare for all possibilities. This requires excessive reservation of resources. Similarly, the required resources and actions cannot be specified completely due to the undetermined choices. Therefore, making sure that the required resources and actions are available in all cases also requires redundant resources and actions.

Formulas involving choices can be classified as follows.

1. A formula may contain no choice, which we call a **definite formula**.

2. A formula may also contain only internal choices without conditions or with conditions that can be satisfied without imposing any constraints on the agents. The agents can decide these internal choices at will. We call this kind of formula a **determinable formula**.

3. The third kind of formulas is the one that contains internal choices with conditions. These conditions put a constraint on the decisions on these internal choices. Such constraints completely determine the decisions on these choices but it is up to agents whether they want to satisfy the constraints or not. Constraints can arise from dependencies, or other goal achievements, or other factors. Formulas of this kind are called **constrained formulas**.

4. The fourth kind corresponds to those formulas that contains indeterminate possibilities
or contains internal choices whose constraints depend on indeterminate possibilities. We call these **risky formulas**.

Note that in some cases, formulas that contain choices whose constraints include agent identifying conditions are also determinable formulas. An agent identifying condition requires a matching of given agent ID with the owner agent’s ID. If there is such a match, then the choice is a free choice to the owner agent and the formula is a determinable formula. Otherwise, the choice is an indeterminate possibility and the formula is a risky formula.

It can be seen that these four types of formulas vary in the degree of certainty of their exact final forms. Moreover, each kind of formulas might be subject to different agent strategies about choices. Specifically, determinable formulas and constrained formulas are subject to strategies about deciding choices at their exact times or in advance. Note that deciding constrained formulas also involves the task of fulfilling the constraints. Risky formulas are subject to strategies concerning with whether to prepare for all possibilities or to make predictions.

We consider some examples of applying agent choice strategies to these four kinds of formulas in the context of goal and base commitment formulas and resource formulas.

Consider goal and base commitment formulas. To fulfill a definite commitment $A^\perp$, the agent needs the resource $A$. To fulfill a determinable commitment such as $[O^n A \& _x 1]^\perp$, the agent can decide the commitment to be $A^\perp$ or $1^\perp$ (in effect, ignoring the commitment) and hence needs $A$ or $1$ respectively. To fulfill a constrained or risky commitment such as $[O^n A \oplus_x 1]^\perp$, there is a certain level of uncertainty involved. If the agent is taking risks, it can predict the commitment as being $A^\perp$ or $1^\perp$ and act accordingly. If the agent wants to be safe, it must prepare for both cases, which means it can not ignore the commitment of $A^\perp$.

Resource formulas are similar. While having a definite resource $A$ is certain, having a determinable resource $[O^n A \& _x 1]$ means that the agent can freely decide to have the resource $A$ or not. A constrained resource $[O^n A \& _x 1]$ with some constraints on $\& _x$ means that the agent must satisfy these constraints for $A$ to result. In those cases where conditions can not be satisfied or involve indeterminate possibilities, constrained resources are treated in the same way as risky resources. A risky resource $[O^n A \oplus_x 1]$ may turn out to be $A$ or 1, which means no resource. Hence, taking $[O^n A \oplus_x 1]$ as $A$ for consideration is a risky strategy.

Moreover, in the context of agent interaction, in order to deal with formulas from other agents, an agent can exercise various strategies depending on the particular kinds of the formulas to determine their final forms. For example, an agent can take a bold strategy
toward a risky resource formula provided by another agent by assuming that the risky formula has a particular desirable form (i.e. some of its indeterminate possibilities have desirable outcomes).

Agent choice strategies can be applied at various stages during an interaction, such as resolving a commitment internally, sending and receiving a request, sending and receiving a proposal, fulfilling a proposal’s conditions and carrying out an agreed proposal. Applications of these strategies remove choices and hence reduce the formula of concern further toward its final form where all the decisions on choices are made.

Furthermore, compound formulas may be arbitrarily complex and contain several choices in them. In many cases, the agents are only interested in some parts of the compound formulas. However, because the final forms of these compound formulas may vary widely, the parts of interest may or may not exist in the final forms. To facilitate agent decision making on strategies to deal with these formulas, we take a “divide and conquer” approach.

Firstly, a mechanism for splitting up a compound formula into subformulas will be provided, as discussed in the next chapter. Secondly, given this ability to separate the parts (sub-formulas) of interest from the rest, we then allow agents to focus on these parts and consider each part together with its retention relationship with the composite formula. Moreover, as discussed in Section 5.2.1, whether a sub-formula is retained in a composite formula can be regarded as the decision of its representative choice. We will discuss further strategies of agents in dealing with representative choices of sub-formulas in Section 6.3.

5.5 Summary

In this chapter, we have addressed issues regarding modeling decisions of agents on internal choices and indeterminate possibilities. A choice calculus that lays a theoretical ground for agent reasoning about various strategies on choices was described. Such strategies include deciding choices in advance or at their associated times, and taking a safe approach or bold approach to indeterminate possibilities.
\[
\begin{array}{ll}
\frac{\Gamma \vdash_{cc} F, \Delta \quad [\vdash \xi \rightarrow L]}{\Gamma \vdash_{cc} F \& n G, \Delta} & \frac{\Gamma \vdash_{cc} G, \Delta \quad [\vdash \xi \rightarrow R]}{\Gamma \vdash_{cc} F \& n G, \Delta} \\
\frac{\Gamma, F \vdash_{cc} \Delta \quad [\vdash \xi \rightarrow L]}{\Gamma, F \oplus n G \vdash_{cc} \Delta} & \frac{\Gamma, G \vdash_{cc} \Delta \quad [\vdash \xi \rightarrow R]}{\Gamma, F \oplus n G \vdash_{cc} \Delta} \\
\frac{\Gamma \vdash_{cc} \bigcirc^n F, \Delta \quad [\vdash \bigcirc^n \xi \rightarrow L]}{\Gamma \vdash_{cc} \bigcirc^n (F \& n G), \Delta} & \frac{\Gamma \vdash_{cc} \bigcirc^n G, \Delta \quad [\vdash \bigcirc^n \xi \rightarrow R]}{\Gamma \vdash_{cc} \bigcirc^n (F \& n G), \Delta} \\
\frac{\Gamma, \bigcirc^n F \vdash_{cc} \Delta \quad [\vdash \bigcirc^n \xi \rightarrow L]}{\Gamma, \bigcirc^n (F \oplus n G) \vdash_{cc} \Delta} & \frac{\Gamma, \bigcirc^n G \vdash_{cc} \Delta \quad [\vdash \bigcirc^n \xi \rightarrow R]}{\Gamma, \bigcirc^n (F \oplus n G) \vdash_{cc} \Delta} \\
\frac{\Gamma \vdash_{cc} F, \Delta \quad [\text{condL}_n]}{\Gamma, F \& n G \vdash_{cc} \Delta} & \frac{\Gamma \vdash_{cc} G, \Delta \quad [\text{condR}_n]}{\Gamma, F \& n G \vdash_{cc} \Delta} \\
\frac{\Gamma \vdash_{cc} F, \Delta \quad [\text{condL}_n]}{\Gamma \vdash_{cc} F \oplus n G, \Delta} & \frac{\Gamma \vdash_{cc} G, \Delta \quad [\text{condR}_n]}{\Gamma \vdash_{cc} F \oplus n G, \Delta} \\
\frac{\Gamma \vdash_{cc} \bigcirc^n F, \Delta \quad [\text{condL}_n]}{\Gamma \vdash_{cc} \bigcirc^n (F \& n G), \Delta} & \frac{\Gamma \vdash_{cc} G, \bigcirc^n \Delta \quad [\text{condR}_n]}{\Gamma \vdash_{cc} \bigcirc^n (F \oplus n G), \Delta} \\
\end{array}
\]

Table 5.1: New Sequent Rules of the Choice Calculus
Chapter 6

Partial and Concurrent Handling of Goals and Resources

In a distributed environment such as a multi-agent system, an agent is likely to interact with several other agents. Often agents achieve one part of their goal in one interaction and other parts in other interactions. Accordingly, goals (or commitments) are likely to be worked on in a distributed and concurrent manner and similarly for resources and actions. Furthermore, the precise allocation of sub-parts to interactions may not be known in advance. It is then important to provide a mechanism for agents to dynamically divide compound goals and base commitments and compound resources and actions.

Moreover, there are some desirable properties for this dynamic division. Firstly, the fulfilled parts of goals or base commitments (or the used part of resources or actions) should be removed from consideration, leaving only those unfulfilled (respectively unused) intact. Secondly, since parts of a compound goal or base commitment (or resource or action) are usually interdependent, updating one should influence the other. Especially in the context of choices, some parts may not only co-exist concurrently but also be mutually exclusive and subject to the agents’ decisions about choices. Hence, when agents make decisions about choices, it is important to ensure that the effects are propagated to all relevant parts.

We will discuss a suitable fragment of TLL in Section 6.1. Based on this fragment, a mechanism to divide formulas of goals or base commitments and resources or actions into multiple sub-formulas dynamically is described in Section 6.2. We also describe how the model of choices and their dependencies described in Chapter 5 can be used to capture dependencies among sub-parts. The relationship between a formula and its constituent parts
is further examined in a resource context. How the agents’ decisions about choices effect the distribution of a formula is examined in Section 6.3. Applications of the splitting up mechanism to agent reasoning about partial handling of goals and base commitments or resources and actions, and to distributed concurrent problem solving are discussed in Section 6.4 and Section 6.5 respectively.

6.1 Restrictions on the Logic Fragment

In this section, we discuss constraints on the fragment of TLL used to enable decomposition of a formula such that the sub-parts are equivalent to the original in terms of resources. In particular, given that the suitable connective for describing the co-existence among sub-parts in TLL is multiplicative conjunction \(\otimes\), we consider equivalence between a compound formula and a multiplicative conjunction of its sub-parts from the perspective of resources.

As discussed in Chapter 4 and especially in Section 4.2.1, our model of agent resources or actions and goals or base commitments makes use of the TLL connectives \(\otimes, \&\), \(\oplus\) (without \(-\rightarrow\)) and temporal operators \(\square, \Diamond\) and \(\bigcirc\). The relationships between the formulars in this fragment makes the decomposition of a compound formula into an equivalent multiplicative conjunction of sub-formulas a non-trivial task. In fact, there are cases where a multiplicative conjunction of sub-parts is not logically equivalent to the original formula. For example,

1. \(\square (A \otimes B) \not\equiv \square A \otimes \square B\)
2. \(\Diamond (A \otimes B) \not\equiv \Diamond A \otimes \Diamond B\)
3. \(\bigcirc (A \otimes B) \not\equiv \bigcirc A \otimes \bigcirc B\)
4. \(\Diamond A \otimes \Diamond B \not\equiv \Diamond (A \otimes B)\)

While the first three cases can be worked around (as discussed in Section 6.2.1), cases such as the fourth one remain a challenging problem.

In order to overcome this, we revisit the model of “anytime” and “sometime”. While “anytime” refers to a particular time point of choice as decided by the agents, “sometime” refers to a time point that is not known and not decided by the agents. As discussed, \(\square\) and \(\Diamond\) are used to intuitively capture these notions. However, this is not the only approach to modeling these notions. Indeed, in the context of discrete time as in TLL, “anytime \(A\)” can be intuitively interpreted as \(\bigcirc^x A\), where \(x\) is a variable whose value is determined by the agents \((x \geq 0)\). “Sometime” \(A\) can be interpreted as \(\bigcirc^n A\), where \(n\) is an unknown number
(n ≥ 0). Hence, the value of x is essentially the outcome of an internal decision by the agent, either now or the next time point or the second next time point, etc. The value of n is essentially the outcome of an indeterminate possibility, either now or the next time point or the second next time point, etc. Such interpretations enable a modeling of “anytime” (respectively “sometime”) as an internal choice (respectively indeterminate possibility) in time. Logically, such modeling is captured as follows:

"anytime" A ≈ A & \(\bigcirc\) (A & \(\bigcirc\) (A & . . . . \(\bigcirc\) A))

which means that agents can choose either A now or A at the next time point or A at the second next time point, etc. It is an internal choice by the agent.

"sometime" A ≈ A ⊕ \(\bigcirc\) (A ⊕ \(\bigcirc\) (A ⊕ . . . . \(\bigcirc\) A))

which means that A possibly occurs now or will possibly occur at the next time point or at the second next time point, etc. When A occurs is an indeterminate possibility to agents.

Such modeling of “anytime” and “sometime”, as compared to using \(\Box\) and \(\bigcirc\), emphasizes the operational aspect, i.e. what agents do when it comes to “anytime” and “sometime”. The modeling shows more explicitly the choices involved at each time point that agent has to deal with as time passes. In addition, as discussed in Chapter 5, agents can exercise various strategies in dealing with choices, their dependencies and constraints. Such explicit enumeration on choices of “anytime” and “sometime” enables attachment of choice constraints and dependencies on individual choices and hence allows agents to reason about them in a richer manner. For example, “anytime A” may be enriched by some conditions as follows:

"anytime" A ≈ A & \(\bigcirc\) \(\bigcirc\) (A & \(\bigcirc\) (A & . . . . \(\bigcirc\) A))

where \(\text{condL} \& 2 = \vdash (\leftrightarrow \& 2 \rightarrow L)\), for some choice \(\& a\).

The attachment of this condition means that although the agent can have A anytime, at the next time point it must decide whether to have A or not consistently with the outcome of the choice \(\& a\) (choosing the same side).

Another example about “sometime A” is that agents can make predictions on some indeterminate possibilities.
Given "sometime" \( A \approx A \oplus_1 \bigcirc (A \oplus_2 \bigcirc (A \oplus_3 \ldots \bigcirc A)) \)

the agent can make a prediction on the choices \( \oplus_2 \) and \( \oplus_4 \) that the outcomes are the right side, which means \( A \) will not result at these times. Hence, "sometime \( A \)" is predicted as having \( A \) sometime, but not at the next time point nor the third next time point.

Furthermore, as in practice most agent interactions occur within a predictable amount of time, it is acceptable to consider agent interactions in a bounded time context. In particular, we do not consider operations, resources and events that have infinite time span but within a limited time range. Taking this as an assumption, the operating time of an agent interaction has a practical limit of \( T \) time points ahead, so that all events take place between now and \( T \). Under this assumption, the intended meaning of \( \square \) is "anytime" from now until \( T \) and respectively \( \Diamond \) is "sometime" between now and \( T \). Hence, the modeling of "anytime" and "sometime" are re-written as:

\[
\begin{align*}
\square A & \approx A \& \bigcirc (A \& \bigcirc (A \& \ldots \bigcirc A)) \\
\Diamond A & \approx A \oplus \bigcirc (A \oplus \bigcirc (A \oplus \ldots \bigcirc A))
\end{align*}
\]

In addition, if we focus on formulas used for goals, base commitments, resources and actions, then the connectives used are \( \otimes, \&, \oplus, \bigcirc, \square, \Diamond \) and negation \( \bot \). Without loss of generality, splitting up a negative compound formula can be done with its corresponding positive form and followed by a negation.

Hence, by making the assumption of bounded time for "sometime" and "any time", the \( \square \) and \( \Diamond \) operators can be replaced by a combination of the connectives \( \&, \oplus \) and \( \bigcirc \). This has many advantages over the use of \( \square \) and \( \Diamond \) discussed above and also enables the equivalence of a formula and a multiplicative conjunction of its split ups, as will be discussed in Section 6.2.1.

The logic fragment used for modeling can then be limited to the connectives \( \otimes, \&, \oplus \) and time operator \( \bigcirc \). The fragment is formally defined below.

**Definition 9.** A formula \( F \) belongs to the logic fragment MCA (Multiplicative Conjunction Additives) iff it is defined by the following grammar:

\[
F ::= A | \bot | \bigcirc F | (F \otimes F) | (F \& F) | (F \oplus F)
\]
where $A$ is an atom.

We define a basic TLL formula as one of the form $\bigodot^n A$, $0 \leq n \leq T$. When $n = 0$, the basic TLL formula is simply $A$. In this fragment, the relationships between basic TLL formulas can be concurrently co-existing or be mutually exclusive, corresponding to the connectives $\otimes$ or $\&$ and $\oplus$ respectively.

### 6.2 Dynamic Division of Formulas

In this section, we investigate how to turn a compound formula in MCA into its sub-parts. For a running example, we follow the example used in Chapter 5. Further to the description of the original example in Section 5.1, we assume that Peter now has an additional goal of having either Chinese or Thai food at the party. Deriving which goal – Chinese food (abbreviated as $C$) or Thai food (abbreviated as $T$) – is an internal choice. Also, Peter adopts the goal of retaining his CD burner. Peter’s goal is then (CD burner is abbreviated as $CDB$, music is abbreviated as $m$)

$$CDB \otimes \bigodot^2 [m \otimes (C \oplus_3 T)]$$

Note that strictly speaking Peter’s goal should be

$$(CDB \otimes \bigodot^2 [m \otimes (C \oplus_3 T)])^{-1}$$

However, for ease of discussion, we keep the negation $-1$ implicit by denoting the positive form above as a goal formula.

Peter can provide neither of the food options, but his friend Ming can make Chinese food and another friend Chaeng can make Thai food. Hence, the overall goal requires Peter to interact with John and David as described in Chapter 5 and also with Ming or Chaeng regarding the provision of food. If this goal is sent as a request to any one of them, none would be able to fulfill the goal in its entirety. Hence, it is important that the goal can be split up and achieved partially via concurrent threads of interaction.

Our approach to splitting up a compound formula into sub-formulas is to determine how to extract a basic TLL formula from a compound one. In other words, we look at how to turn a compound formula into a multiplicative conjunction of basic TLL formulas, and then how to remove a particular basic TLL formula from it. This approach can be extended to the general case of splitting up a compound formula w.r.t. an arbitrary sub-formula. Indeed,
by treating an arbitrary sub-formula as a single indivisible unit like a basic TLL formula, we can apply the same mechanism to separate the sub-formula from the rest of the formula. Hence, for simplicity of discussion and without losing generality, we focus on the case when the sub-formula is a basic TLL formula.

A basic TLL formula $A$ which is a sub-formula of a compound formula $\Gamma$ may or may not be retained in $\Gamma$ after all the outcomes of choices in $\Gamma$ are determined. Hence, when we separate $A$ from $\Gamma$, it is important to transform the retention relationship of $A$ with $\Gamma$ into the split up that contains $A$.

Let $\Gamma$ be a formula in the fragment MCA that contains $A$. We split up $\Gamma$ into $\hat{A}$, which contains $A$, and $\hat{\Gamma} - \hat{A}$, which is the remainder.

The intuition behind $\hat{\Gamma} - \hat{A}$ is that it is the formula $\Gamma$ having undergone a single removal or substitution of the occurrence of $A$ by 1 while the rest is kept unchanged. Specifically, $\hat{\Gamma} - \hat{A}$ no longer contains the $A$ but it still keeps all the choices that are related to the retention of $A$ in $\Gamma$. In its place in the structure of $\Gamma$, if the immediate connective of $A$ or $\bigodot^n A$ (for some $n$) refers to a choice then $A$ is replaced by 1, whose occurrence in a context of multiplicative conjunction does not matter. 1 is used for technical reasons in order to retain the structure of the choice. Otherwise, if the immediate connective of $A$ or $\bigodot^n A$ is a multiplicative conjunction, then $A$ or $\bigodot^n A$ is removed. All other choices related to $A$ are kept.

In the example, let $\Gamma = CDB \otimes \bigodot^2[m \otimes (C \oplus_3 T)]$. The structure of $\Gamma$ is depicted in Figure 6.1.

$$\hat{\Gamma} - \hat{C}$$

is obtained by replacing $C$ in the structure of $\Gamma$ by 1, as in Figure 6.2.

In general, $\hat{\Gamma} - \hat{A}$ is formed by performing one substitution in $\hat{\Gamma}$ at the point where $A$ is in $\Gamma$. The substitution is defined as below.

**Definition 10.** *(Immediate Sub-formula)*
Let $\Gamma$ be a formula in the fragment MCA (definition 9) and $A$ be a sub-formula of $\Gamma$. A formula $S$ is called the immediate sub-formula in $\Gamma$ that contains $A$ iff $S$ is a sub-formula of $\Gamma$ and $S$ is formed by the grammar:

$$S ::= \bigcirc A|A \otimes A|A \oplus A|A \& A|\bigcirc S$$

where $\Delta$ is a sub-formula of $\Gamma$.

**Definition 11.** ($\Gamma \rightarrow A$)

Let $\Gamma$ be a formula in the fragment MCA (definition 9) and $A$ be a sub-formula of $\Gamma$. Let $S$ be the immediate sub-formula in $\Gamma$ that contains $A$. Let $\mapsto$ denote a substitution of formulas. $\Gamma \rightarrow A$ is defined as follows according to the possible structures of $S$.

- **case** $S = A$: then $A \mapsto 1$
- **case** $S = \bigcirc A$: then $S \mapsto \bigcirc 1$
- **case** $S = \bigcirc^x A \otimes \Delta$: then $S \mapsto \Delta$
- **case** $S = \bigcirc^x A \oplus_m \Delta$: then $S \mapsto \bigcirc^x 1 \oplus_m \Delta$
- **case** $S = \bigcirc^x A \&_n \Delta$: then $S \mapsto \bigcirc^n 1 \&_n \Delta$

where $n, m$ are some numbers representing IDs of choices and $\Delta$ is a sub-formula in $\Gamma$.

In other words, $\Gamma \rightarrow A$ results from $\Gamma$ by replacing the occurrence of $A$ by 1 and then applying the equivalence $1 \otimes \Delta \equiv \Delta$, so that $\bigcirc^x A \otimes \Delta \mapsto \Delta$ where the substitution takes place in $\Gamma$.

The intuition behind the formulation of $\hat{A}$ is that $\hat{A}$ not only represents $A$ but also captures how $A$ is retained in $\hat{A}$ the same way as in $\Gamma$. Specifically, the choices in $\Gamma$ and
the sequence of their decisions that make $A$ retained in $\Gamma$ also make $A$ retained in $\hat{\Lambda}$ and vice versa. Hence, $\hat{\Lambda}$ should contain all the choices related to $A$ in $\Gamma$ organized in a similar structure. Also, if $A$ is not retained in $\Gamma$, then formula $\hat{\Lambda}$ should reflect this, i.e. the neutral element $1$.

We look closely at how $\hat{\Lambda}$ is constructed by reconsidering the formulation of $\Gamma - A$. Rather than substituting $A$ or $\bigcirc^n A$ (for some $n$) as one side of a choice by $1$, we keep it and substituting the other side by $1$. Rather than removing $A$ or $\bigcirc^n A$ as one side of a multiplicative conjunction, we remove the other side. Regarding the rest of the formula, $\hat{\Lambda}$ should be different to $\Gamma - A$ in that all the sides of choices that do not contain $A$ are represented by $1$ and all the sides of a multiplicative conjunction that do not contain $A$ are removed. This makes $\hat{\Lambda}$ become $\bigcirc^m 1$ (for some $m$) when $A$ is not retained from $\Gamma$. As an example, we consider how $\hat{\Lambda}$ is formed from $\Gamma = CDB \bigotimes \bigcirc^2 [m \bigotimes (C \bigoplus T)]$ in figure 6.3.

![Figure 6.3: Structure of $\hat{\Lambda}$](image)

Moreover, after determining all the outcomes of choices in $\hat{\Lambda}$, $\hat{\Lambda}$ should become either $\bigcirc^n A$ or $\bigcirc^m 1$, for some number $n$, $m$. Hence, the formula of $\hat{\Lambda}$ has the following grammar

$$F ::= A | 1 \bigotimes (F \& F) \bigoplus (F \bigoplus F)$$

Note that the grammar above is different from the grammar of the fragment MCA in that $\perp$ and $\otimes$ do not occur.

We then define $\hat{\Lambda}$ by firstly defining how $\hat{\Lambda}$ is obtained in the splitting up of $\Gamma$.

**Definition 12.** ($\text{SPLITUP}(\Gamma, A)$)

Let $\Gamma$ be a formula in the fragment MCA, and $A$ be a sub-formula in $\Gamma$. Where appropriate, let $\Gamma'$ and $\Delta$ be sub-formulas of $\Gamma$ that contains and does not contain $A$ respectively.

$\text{SPLITUP}(\Gamma, A)$ is defined recursively as a transformation process on $\Gamma$ according to the structure of $\Gamma$ as follows.

- if $\Gamma = A$, then $\text{SPLITUP}(\Gamma, A) = A$
Dynamic Division of Formulas

- if $\Gamma = \circ \Gamma'$, then $\text{SPLITUP}(\Gamma, A) = \circ \text{SPLITUP}(\Gamma', A)$
- if $\Gamma = \Gamma' \otimes \Delta$, then $\text{SPLITUP}(\Gamma, A) = \text{SPLITUP}(\Gamma', A)$
- if $\Gamma = \Gamma' \& \Delta$, then $\text{SPLITUP}(\Gamma, A) = \text{SPLITUP}(\Gamma', A) \& \Delta$
- if $\Gamma = \Gamma' \oplus_m \Delta$, then $\text{SPLITUP}(\Gamma, A) = \text{SPLITUP}(\Gamma', A) \oplus_m 1$

where $n, m$ are some numbers representing IDs of choices.

The definition of $\hat{A}$ is then straightforward.

**Definition 13. ($\hat{A}$)**

Let $\Gamma$ be a formula in the fragment MCA, and $A$ be a sub-formula in $\Gamma$. $\hat{A}$ is defined as the outcome of the transformation process $\text{SPLITUP}(\Gamma, A)$.

$$\hat{A} = \text{SPLITUP}(\Gamma, A)$$

Another view is that $\hat{A}$ is obtained by recursively replacing formulas that rest on the other side of connective (to the formula that contains $A$) by 1 if the connective is $\oplus$ or $\&$ and remove the formulas if the connective is $\circ$.

Returning to our running example, we consider how Peter’s goal formula $G = CDB \otimes \circ^2[m \otimes (C \oplus_3 T)]$ can be split up. As we want the subgoal $C$ to be attempted by Ming, we split the goal with respect to $C$:

$$[G \sim C] = CDB \otimes \circ^2[m \otimes (1 \oplus_4 T)]$$

and

$$\hat{C} = \circ^2(C \oplus_5 1).$$

Furthermore, the sub-goal $T$ should also be separated for an interaction with Chaeng. Subsequently, $\hat{G} \sim C$ is split into:

$$\hat{T} \text{ (of } \hat{G} \sim C \text{) is } \circ^2(1 \oplus_6 T).$$

and $$[G \sim C \sim T] = CDB \otimes \circ^2m.$$

The choices $\oplus_4$, $\oplus_5$ and $\oplus_6$ will then be determined as the same.
It can be seen from the definitions of $\Gamma \rightarrow \hat{A}$ and $\hat{A}$ that choices that are related to the retention of $A$ in $\Gamma$ are copied to $\hat{A}$ and retained in $\Gamma \rightarrow \hat{A}$. Indeed, outcomes of these choices in the two split ups must be consistent. Hence, decisions made on these choices in one split up must also be followed in the other. In other words, the sequence of decisions that would apply to these choices in $\Gamma$ now applies to the choices in both $\hat{A}$ and $\Gamma \rightarrow \hat{A}$.

We consider in detail such dependencies on choices in $\hat{A}$ and $\Gamma \rightarrow \hat{A}$ that are related to the retention of $A$ in $\Gamma$ for one of the cases. Assume that $\Gamma \rightarrow \hat{A}$ is obtained from $\Gamma$ by replacing in $\Gamma$ the sub-formula $A \&_n \Delta$ by $1 \&_n \Delta$. From the definition of $\hat{A}$, the choice $\&_n$ (assumed as internal choice) is also present in $\hat{A}$ as $A \&_n 1$. Hence, if the choice $\&_n$ is made in $\hat{A}$ to be left chosen for $A$ to be retained in $\hat{A}$ then as a result, the choice $1 \&_n \Delta$ in $\Gamma \rightarrow \hat{A}$ has the outcome of being left chosen and $\Delta$ is not retained in $\Gamma \rightarrow \hat{A}$. This correspondence ensures that $A$ and $\Delta$ are still mutually exclusive even they are in two separate formulas. Similarly, if the choice $1 \&_n \Delta$ in $\Gamma \rightarrow \hat{A}$ is right chosen, which means that $\Delta$ is retained in $\Gamma \rightarrow \hat{A}$ then decision on the choice $A \&_n 1$ in $\hat{A}$ must follow (choosing the right side). Moreover, instead of an internal choice $\&_n$, if the case is an indeterminate possibility $\oplus_n$ and it turns out that $A$ is retained in $\hat{A}$ then the external choice $1 \oplus \Delta$ in $\Gamma \rightarrow \hat{A}$ also has the outcome that $\Delta$ is not retained in $\Gamma \rightarrow \hat{A}$ and vice versa.

A summary on the inter-dependencies between outcomes of related choices in the split ups is in Table 6.1. The leftmost column describes the cases of the substitutions in $\Gamma$ that lead to $\Gamma \rightarrow \hat{A}$. Other cases of substitutions do not directly involve a choice and hence are not listed. The two middle columns describe the corresponding choices in the two split ups. The rightmost column shows outcomes of the choice and what is retained from these choices in $\Gamma \rightarrow \hat{A}$ and $\hat{A}$ respectively.

<table>
<thead>
<tr>
<th>$\Gamma$</th>
<th>$\Gamma \rightarrow \hat{A}$</th>
<th>$\hat{A}$</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A &amp;_1 \Delta$</td>
<td>$1 &amp;_1 \Delta$</td>
<td>$A &amp;_1 1$</td>
<td>$&amp;_1 \Rightarrow L; 1; A$</td>
</tr>
<tr>
<td>$A &amp;_1 \Delta$</td>
<td>$1 &amp;_1 \Delta$</td>
<td>$A &amp;_1 1$</td>
<td>$&amp;_1 \Rightarrow R; \Delta; 1$</td>
</tr>
<tr>
<td>$A \oplus_2 \Delta$</td>
<td>$1 \oplus_2 \Delta$</td>
<td>$A \oplus_2 1$</td>
<td>$\oplus_2 \Rightarrow L; 1; A$</td>
</tr>
<tr>
<td>$A \oplus_2 \Delta$</td>
<td>$1 \oplus_2 \Delta$</td>
<td>$A \oplus_2 1$</td>
<td>$\oplus_2 \Rightarrow R; \Delta; 1$</td>
</tr>
</tbody>
</table>

*Table 6.1: Summary of Inter-dependencies on choices among split ups*

Referring to our running example, between the split ups, consistency in the corresponding choices must be maintained. In particular, the choices $\oplus_4$ and $\oplus_5$ must share the same outcomes with the choice $\oplus_3$. Moreover, the outcome of the choice $\oplus_6$ must also be the same as
that of the choice $\oplus_5$ and hence the same as that of $\oplus_3$. More precisely, the split ups are then

$$[G - C] = CDB \otimes \bigcirc^2[m \otimes (1 \oplus_3 T)],$$

$$\hat{C} = \bigcirc^2(C \oplus_3 1),$$

$$\hat{T} \text{ (of } G - C) = \bigcirc^2(1 \oplus_3 T), \text{ and}$$

$$[G - C - T] = CDB \otimes \bigcirc^2 m.$$

### 6.2.1 Equivalence between a Formula and Its Split Ups

Given such a mechanism to split up a compound formula into sub-formulas and maintain the consistency between related choices, it is then important to establish the relationship between the formula and its split ups. In particular, we establish formal results for all the split up formulas $\approx$ original formula.

The following theorem demonstrates that the two split ups can derive the original formula in the choice calculus.

**Theorem 6.2.1.** Let $\Gamma$ be a formula in the fragment MCA that contains $A$. Let $\Gamma$ be split up into $\hat{A}$ and $\hat{\Gamma - A}$ w.r.t. $A$, where $\hat{A}$ contains $A$ and $\hat{\Gamma - A}$ is the remainder. Then

$$\hat{A}, \hat{\Gamma - A} \vdash_{cc} \Gamma$$

**Proof:** the proof is by induction on the size of $\Gamma$.

**Base step:** $\Gamma = A$. We need to prove $A, 1 \vdash A$, which is obvious.

**Induction step:**
Assume the hypothesis is true for $\Gamma$ of size $n$, denoted as $\Gamma^n$.

$$\hat{A}^n, [\Gamma^n - A]^n \vdash_{cc} \Gamma^n.$$
\( \bigcirc^2 \Gamma^n, \Gamma^n \otimes B, \Gamma^n \&_1 B \) and, \( \Gamma^n \oplus_2 B \).

Note that for the first case, \( \widehat{A}^{n+1} = \bigcirc^2 A^n \), for the second case, \( \widehat{A}^{n+1} = \widehat{A}^n \) and for the last two cases \( \widehat{A}^{n+1} = \widehat{A^n} \&_1 1 \) and \( \widehat{A}^{n+1} = \widehat{A^n} \oplus_2 1 \) respectively.

We will consider each case of the structure of \( \Gamma_{n+1} \). In each case, we will reduce the \((n+1)\) case to the hypothesis of \( n \) case by providing the relevant proof steps. Note that when an inference rule is labeled with \([n]\), it means several applications of the rule.

**Case:** \( \Gamma_{n+1} = \bigcirc^2 \Gamma^n \).

We need to prove \( \bigcirc^2 \widehat{A^n}, \bigcirc^2 [\widehat{\Gamma^n - A}]^n \vdash_{cc} \bigcirc^2 \Gamma^n \) based on the hypothesis \( \widehat{A^n}, [\widehat{\Gamma^n - A}]^n \vdash_{cc} \Gamma^n \). It is straightforward to show that

\[
\frac{\widehat{A^n}, [\widehat{\Gamma^n - A}]^n \vdash_{cc} \Gamma^n}{\bigcirc^2 \widehat{A^n}, \bigcirc^2 [\widehat{\Gamma^n - A}]^n \vdash_{cc} \bigcirc^2 \Gamma^n}
\]

Other cases for \( \Gamma_{n+1} \) are proved similarly.

**Case:** \( \Gamma_{n+1} = \Gamma^n \otimes B \).

We need to prove \( \widehat{A^n}, [\widehat{\Gamma^n - A}]^n \otimes B \vdash_{cc} \Gamma^n \otimes B \). This is straightforward, as follows.

\[
\frac{\widehat{A^n}, [\widehat{\Gamma^n - A}]^n \otimes B \vdash_{cc} \Gamma^n \otimes B}{\Gamma^n \otimes B \vdash_{cc} \bigcirc^2 \widehat{A^n}, \bigcirc^2 [\widehat{\Gamma^n - A}]^n \vdash_{cc} \bigcirc^2 \Gamma^n}
\]

**Case:** \( \Gamma_{n+1} = \Gamma^n \&_1 B \).

We need to prove \( \widehat{A^n} \&_1 [\widehat{\Gamma^n - A}]^n \&_1 B \vdash_{cc} \Gamma^n \&_1 B \). \&_1 corresponds to an internal choice whose conditions for being left chosen and right chosen are \( condL_1 \) and \( condR_1 \) respectively. We have

\[
\frac{\widehat{A^n}, [\widehat{\Gamma^n - A}]^n \vdash_{cc} \Gamma^n [condL_1]}{\widehat{A^n} \&_1 1, [\widehat{\Gamma^n - A}]^n \&_1 B \vdash_{cc} \Gamma^n \&_1 B}
\]

\[
\frac{B \vdash \Gamma^n \&_1 [condR_1]}{B, 1 \vdash \Gamma^n \&_1 B [condR_1]}
\]

\[
\frac{\widehat{A^n} \&_1 1, [\widehat{\Gamma^n - A}]^n \&_1 B \vdash_{cc} \Gamma^n \&_1 B}{\&_1}
\]

All the cases of the outcome of the internal choice that corresponds to \&_1 are proved.

**Case:** \( \Gamma_{n+1} = \Gamma^n \oplus_2 B \).
We need to prove $\widehat{A^n \oplus_2 1}, [\Gamma^n - A]^n \oplus_2 B \vdash_{cc} \Gamma^n \oplus_2 B$ ($\oplus_2$ is an indeterminate possibility). We have

$$\frac{\widehat{A^n, [\Gamma^n - A]^n \vdash_{cc} \Gamma^n \vdash_{cc} L}}{\widehat{A^n \oplus_2 1, [\Gamma^n - A]^n \oplus_2 B \vdash_{cc} \Gamma^n \oplus_2 B}} \quad \oplus_2$$

or

$$\frac{1, B \vdash_{cc} B \vdash_{cc} R \quad 1L}{\widehat{A^n \oplus_2 1, [\Gamma^n - A]^n \oplus_2 B \vdash_{cc} \Gamma^n \oplus_2 B}} \quad \oplus_2$$

All the cases of the outcome of the indeterminate possibility that corresponds to $\oplus_2$ are proved.

Hence, for all the cases of $\Gamma_{n+1}$, we can establish that

$$\widehat{A^{n+1}, [\Gamma^{n+1} - A]^{n+1} \vdash_{cc} \Gamma^{n+1}}$$

By induction, we can conclude $\widehat{A, \Gamma - A} \vdash \Gamma$.

We now come to see if a multiplicative conjunction of the split ups ($\widehat{A \otimes \Gamma - A}$) is derivative from the original formula $\Gamma$. Specifically, we need to determine if the sequent:

$$\Gamma \vdash \widehat{A \otimes \Gamma - A}$$

holds in the choice calculus.

Consider $\Gamma = \circ (A \otimes B)$. Then $\widehat{A} = \circ A$ and $\widehat{\Gamma - A} = \circ B$. The sequent does not hold for this case as

$$\circ (A \otimes B) \not\vdash_{cc} \circ A \otimes \circ B.$$  

Hence, we need to determine a weaker derivative property from $\Gamma$ to its split ups. We firstly define other forms of the split ups.

**Definition 14.** ($\widehat{A^\perp}$)

Let $\Gamma$ be a formula in the fragment MCA (definition 9) and $A$ be a sub-formula of $\Gamma$. Let $\widehat{A}$ be a split of $\Gamma$ that contains $A$. Then $\widehat{A^\perp}$ is defined as the result of replacing the single copy of $A$ in $\widehat{A}$ by $A^\perp$. 
**Definition 15. \((\Gamma ^* - A)\)**

Let \(\Gamma\) be a formula in the fragment MCA (definition 9) and \(A\) be a sub-formula of \(\Gamma\). Let \((\Gamma ^* - A)\) be a split of of \(\Gamma\) that does not contain \(A\). Then \(\Gamma ^* - A\) is defined as \(\Gamma ^* - A\) except that where \(A\) is replaced by 1 in \(\Gamma ^* - A\), it is replaced by \(\bot\) instead.

We have the following theorem.

**Theorem 6.2.2.** Let \(\Gamma\) be a formula in the fragment MCA that contains \(A\). Let \(\Gamma\) be split up into \(\hat{A}\) and \(\Gamma - A\) w.r.t. \(A\), where \(\hat{A}\) contains \(A\) and \(\Gamma - A\) is the remainder. Then

\[
\Gamma, \hat{A} \vdash_{cc} \Gamma ^* - A.
\]

The theorem means that using (a multiplicative conjunction of) \(\Gamma\) and \(\hat{A}\), we can derive \(\Gamma ^* - A\).

**Proof:** by induction on the size of \(\Gamma\).

**Base step:** \(\Gamma = A\).

We need to prove \(A \otimes A^\bot \vdash_{cc} \bot\) which trivially holds.

**Induction step:**

Assume the hypothesis is true for \(\Gamma\) of size \(n\), denoted as \(\Gamma^n\), so that

\[
\Gamma^n, (A^\bot)^n \vdash_{cc} ^*\Gamma^n - A^n.
\]

We need to prove for the case of \(\Gamma\) of size \(n + 1\), denoted as \(\Gamma^{n+1}\).

Possible structures of \(\Gamma^{n+1}\) of size \(n + 1\) are

\[
\bigcirc^z \Gamma^n, \Gamma^n \otimes B, \Gamma^n \&_1 B, \text{ and } \Gamma^n \oplus_2 B.
\]

Note that for the first case, \((A^\bot)^{n+1} = \bigcirc^z (A^\bot)^n\), for the second case, \((A^\bot)^{n+1} = (A^\bot)^n\) and for the last two cases, \((A^\bot)^{n+1} = (A^\bot)^n \&_1 1\) and \((A^\bot)^{n+1} = (A^\bot)^n \oplus_2 1\) respectively.

We will consider each case of the structure of \(\Gamma^{n+1}\). In each case, we will reduce the \((n+1)\) case to the hypothesis of \(n\) case by providing the relevant proof steps.

**Case:** \(\Gamma^{n+1} = \bigcirc^z \Gamma^n\).

We need to prove \(\bigcirc^z \Gamma^n, \bigcirc^z (A^\bot)^n \vdash_{cc} \bigcirc^z \Gamma^n - A^n\) based on the hypothesis \(\Gamma^n, (A^\bot)^n \vdash_{cc} ^*\Gamma^n - A^n\). This is straightforward, as below.
Other cases of the structure of $\Gamma_{n+1}$ are proved similarly.

**Case:** $\Gamma_{n+1} = \Gamma^n \otimes B$.
We need to prove $\Gamma^n \otimes B, (A \perp)^n \vdash \Gamma^n - A \otimes \Gamma^n \otimes B$. This is straightforward, as below.

$$
\frac{\Gamma^n, (A \perp)^n \vdash \Gamma^n - A \otimes B \otimes \Gamma^n \otimes B}{\Gamma^n \otimes B, (A \perp)^n \vdash \Gamma^n - A \otimes \Gamma^n \otimes B} \otimes L
$$

**Case:** $\Gamma_{n+1} = \Gamma^n \&_1 B$.
We need to prove $\Gamma^n \&_1 B, (A \perp)^n \&_1 1 \vdash \Gamma^n - A \otimes B \&_1 B$. \&_1 corresponds to an internal choice whose conditions for being left chosen and right chosen are condL and condR respectively. We have

$$
\frac{\Gamma^n, (A \perp)^n \vdash \Gamma^n - A \otimes \Gamma^n \&_1 B}{\Gamma^n \&_1 B, (A \perp)^n \&_1 1 \vdash \Gamma^n - A \otimes \Gamma^n \&_1 B} \otimes L
$$

or

$$
\frac{B \vdash B \quad [\text{condR}_1]}{B, 1 \vdash B \quad [\text{condR}_1] \otimes L} \&_1 1 L
\frac{\Gamma^n \&_1 B, (A \perp)^n \&_1 1 \vdash \Gamma^n - A \otimes \Gamma^n \&_1 B}{\Gamma^n \&_1 B, (A \perp)^n \&_1 1 \vdash \Gamma^n - A \otimes \Gamma^n \&_1 B} \otimes L
$$

All the cases of the outcome of the internal choice that corresponds to \&_1 are proved.

**Case:** $\Gamma_{n+1} = \Gamma^n \oplus_2 B$.
We need to prove $\Gamma^n \oplus_2 B, (A \perp)^n \oplus_2 1 \vdash \Gamma^n - A \otimes \Gamma^n \oplus_2 B$ ($\oplus_2$ is an indeterminate possibility). This is straightforward as below

$$
\frac{\Gamma^n, (A \perp)^n \vdash \Gamma^n - A \otimes \Gamma^n \oplus_2 L}{\Gamma^n \oplus_2 B, (A \perp)^n \oplus_2 1 \vdash \Gamma^n - A \otimes \Gamma^n \oplus_2 B} \oplus_2
$$

or

$$
\frac{B \vdash B \quad \boxed{\oplus_2}}{B, 1 \vdash B \quad \boxed{\oplus_2} \otimes L} \oplus_2
\frac{\Gamma^n \oplus_2 B, (A \perp)^n \oplus_2 1 \vdash \Gamma^n - A \otimes \Gamma^n \oplus_2 B}{\Gamma^n \oplus_2 B, (A \perp)^n \oplus_2 1 \vdash \Gamma^n - A \otimes \Gamma^n \oplus_2 B} \otimes L
$$
All the cases of the outcome of the indeterminate possibility that corresponds to $\oplus_2$ are proved.

Hence, for all the cases of $\Gamma_{n+1}$, we can establish that

$$\Gamma_{n+1}, (\widehat{A^\perp})_{n+1} \vdash_{cc} [\Gamma_{n+1} - A]_{n+1}$$

By induction, we can conclude $\Gamma, \widehat{A^\perp} \vdash_{cc} \Gamma - A$ \hfill $\square$

The result below then follows immediately.

**Corollary 6.2.3.** $\Gamma, \widehat{A^\perp}, \widehat{A} \vdash_{cc} \Gamma - A \otimes \widehat{A}$.

We further consider simplifying $\widehat{A^\perp} \otimes \widehat{A}$ in a resource conscious context. As they have the same structure except where $A$ occurs in $\widehat{A}$, $A^\perp$ occurs in $\widehat{A}$. This also means that after all choices are determined, $\widehat{A}$ and $\widehat{A^\perp}$ can be either $\circ^n A$ and $\circ^n A^\perp$ respectively or $\circ^n 1$ for some $n, m \geq 0$.

Consider the case where they become $\circ^n A$ and $\circ^n A^\perp$. We have $\circ^n A \otimes \circ^n A^\perp \vdash \bot$ but also $\circ^n A \otimes \circ^n A^\perp \vdash \bot$ because:

$$
\begin{align*}
\frac{\circ^n A \vdash \circ^n A}{A \vdash A} & \quad \circ \rightarrow \circ \\
\frac{\circ^n A \vdash \bot, \circ^n A}{\circ^n A, \circ^n A^\perp \vdash \bot} & \quad \bot \rightarrow \bot \\
\frac{\circ^n A \vdash \bot}{\circ^n A \otimes \circ^n A^\perp \vdash \bot} & \quad \otimes \rightarrow \bot
\end{align*}
$$

which means that $\circ^n A^\perp$ can also be matched with $\circ^n A$ to produce $\bot$. Hence, in our modeling context, while $\circ^n A$ is regarded as what is provided, $\circ^n A^\perp$ can be regarded as what is required, in the same way as $\circ^n A^\perp$. More generally, in this case, $\widehat{A}$ and $\widehat{A^\perp}$ also have similar dual relationship as described in the following theorem.

**Theorem 6.2.4.** Let $\Gamma$ be a formula in the fragment MCA, and $A$ be a sub-formula of $\Gamma$. Let $\widehat{A}$ be a split up of $\Gamma$ that contains $A$ and $\widehat{A^\perp}$ be the result of replacing the exact $A$ in the structure of $\widehat{A}$ by $A^\perp$. If $A$ and $A^\perp$ are chosen in all the choices in $\widehat{A}$ and $\widehat{A^\perp}$ respectively, then the sequent

$$\widehat{A} \otimes \widehat{A^\perp} \vdash$$

is provable in TLL.
Note that the theorem holds under standard TLL sequent calculus.

**Proof:** by induction on the length of $\hat{A}$.

**Base step:** $\hat{A} = A$, hence $A^\perp = A^\perp$. We have $\hat{A} \otimes A^\perp \vdash$ is $A \otimes A^\perp \vdash$, which is an axiom.

**Induction step:**

We assume that the hypothesis is true for the case of $\hat{A}$ of length $n$.

**Hypothesis:** the sequent $\hat{A}^n \otimes (\hat{A}^\perp)^n \vdash$ holds if $A$ and $A^\perp$ are chosen in all the choices in $\hat{A}^n$ and $(A^\perp)^n$ respectively.

For the case $(n+1)$, possible structures of $\hat{A}^{n+1}$ are

$\circ^n\hat{A}^n$, $\hat{A}^n \&_1 1$ and $\hat{A}^n \oplus_2 1$.

In the last two cases of $\hat{A}^{n+1}$, one more choice is introduced as compared to $\hat{A}^n$.

Correspondingly, possible structures of $(\hat{A}^\perp)^{n+1}$ are

$\circ^n(\hat{A}^\perp)^n$, $(\hat{A}^\perp)^n \&_1 1$ and $(\hat{A}^\perp)^n \oplus_2 1$.

We need to prove that $\hat{A}^{n+1} \otimes (\hat{A}^\perp)^{n+1} \vdash$ holds when $A$ and $A^\perp$ are chosen in all the choices in $\hat{A}^{n+1}$ and $(\hat{A}^\perp)^{n+1}$ respectively.

**Case** $\hat{A}^{n+1} = \circ^n\hat{A}^n$: then also $(\hat{A}^\perp)^{n+1} = \circ^n(\hat{A}^\perp)^n$.

$$
\frac{A^n, (A^\perp)^n \vdash}{(A^\perp)^n \vdash [\hat{A}^n]^\perp} \quad \perp L
\frac{(A^\perp)^n \vdash [\hat{A}^n]^\perp}{\circ^n(A^\perp)^n, \circ^n[A^n]^\perp \vdash} \quad \perp R
\frac{\circ^n\hat{A}^n \otimes \circ^n(A^\perp)^n \vdash}{A^{n+1} \otimes (\hat{A}^\perp)^{n+1} \vdash}
$$

**Case** $\hat{A}^{n+1} = \hat{A}^n \&_1 1$: then also $(\hat{A}^\perp)^{n+1} = (\hat{A}^\perp)^n \&_1 1$. $\&_1$ corresponds to an internal choice whose conditions for being left chosen and right chosen are $\text{cond}L_1$ and $\text{cond}R_1$ respectively. Note that for $A$ and $A^\perp$ to be chosen, $\&_1$ must be left chosen, i.e. $[\text{cond}L_1]$ holds.
\[ \frac{A^n, (A^\perp)^n}{A^n \&_1 [\text{cond} L_1]} \quad \&_1 \]
\[ \frac{A^n \&_1 [(A^\perp)^n \&_1]}{A^n \&_1 \otimes [(A^\perp)^n \&_1]} \quad \otimes L \]
\[ \frac{A^n \&_1 \otimes [(A^\perp)^n \&_1]}{A^{n+1} \otimes (A^\perp)^{n+1}} \]

**Case** \( A^{n+1} = A^n \oplus_2 1 \): then also \( (A^\perp)^{n+1} = (A^\perp)^n \oplus_2 1 \) (\( \oplus_2 \) is an indeterminate possibility). Note that for \( A \) and \( A^\perp \) to be chosen, the outcome of \( \oplus_2 \) must be left chosen, i.e. \( \oplus_2 [\text{left} L] \).

\[ \frac{A^n, (A^\perp)^n}{A^n \oplus_2 [\text{left} L]} \quad \oplus_2 L \]
\[ \frac{A^n \oplus_2 [(A^\perp)^n \oplus_2]}{A^n \oplus_2 \otimes [(A^\perp)^n \oplus_2]} \quad \otimes L \]
\[ \frac{A^n \oplus_2 \otimes [(A^\perp)^n \oplus_2]}{A^{n+1} \otimes (A^\perp)^{n+1}} \]

Hence, by induction, the theorem is proved. \( \square \)

Hence, when \( A \) and \( A^\perp \) are chosen in all the choices, we have \( \widehat{A} \otimes \widehat{A^\perp} \vdash \). Otherwise, \( \widehat{A} \otimes \widehat{A^\perp} \) becomes \( \bigcirc^m 1 \otimes \bigcirc^m 1 \) for some \( m \geq 0 \). In other words, the concurrent presence of \( \widehat{A} \) and \( \widehat{A^\perp} \) together does not produce anything in terms of resources and actions.

We consider further in the context of a sequent \( \Gamma, \widehat{A}, \widehat{A^\perp} \vdash \star \Gamma \rightarrow A \otimes \widehat{A} \) (Corollary 6.2.3). Given that, \( \widehat{A} \otimes \widehat{A^\perp} \) does not produce anything in terms of resources and actions, the resources and actions required to produce \( \star \Gamma \rightarrow A \otimes \widehat{A} \) are essentially \( \Gamma \). In other words, the concurrent presence of \( \widehat{A} \) and \( \widehat{A^\perp} \) acts like a catalyst and can be ignored in terms of resource or action requirements. Hence, in terms of resources and actions requirements, the sequent

\[ \Gamma, \widehat{A}, \widehat{A^\perp} \vdash \star \Gamma \rightarrow A \otimes \widehat{A} \]

is essentially

\[ \Gamma \vdash \star \Gamma \rightarrow A \otimes \widehat{A}. \]

Consider our running example. The goal of Peter (\( \Gamma \)) is split up w.r.t. \( C \) to yield

\[ \widehat{C} = \bigcirc^2 (C \oplus_3 1), \]

\[ [\widehat{G} - \widehat{C}] = CDB \otimes \bigcirc^2 [m \otimes (1 \oplus_3 T)] \]
We then have the property $\hat{C}, \hat{G} \vdash C \!\! \vdash \!\! C$ holds:

\[
\begin{align*}
C \vdash \!\! \vdash \!\! C & \vdash \!\! \vdash \!\! L \\
C, 1 \vdash \!\! \vdash \!\! C & \vdash \!\! \vdash \!\! L \\
C \oplus_3 1, 1 \oplus_3 T & \vdash \!\! \vdash \!\! (C \oplus_3 T) \quad m \vdash \!\! \vdash \!\! m \oplus_3 \quad \oplus_3 \\
C \oplus_3 1, m \oplus (1 \oplus_3 T) & \vdash \!\! \vdash \!\! m \otimes (C \oplus_3 T) \quad \otimes_L \\
\otimes^2(C \oplus_3 1), \otimes^2[m \otimes (1 \oplus_3 T)] & \vdash \!\! \vdash \!\! \otimes^2[m \otimes (C \oplus_3 T)] \quad CDB \vdash \!\! \vdash \!\! CDB \quad \otimes_R \\
\otimes^2(C \oplus_3 1), CDB, \otimes^2[m \otimes (1 \oplus_3 T)] & \vdash \!\! \vdash \!\! CDB \otimes \otimes^2[m \otimes (C \oplus_3 T)] \\
\otimes^2(C \oplus_3 1), CDB \otimes \otimes^2[m \otimes (1 \oplus_3 T)] & \vdash \!\! \vdash \!\! CDB \otimes \otimes^2[m \otimes (C \oplus_3 T)] \\
\hat{C}, \hat{G} \vdash \!\! \vdash \!\! C & \vdash \!\! \vdash \!\! G \\
\end{align*}
\]

or at the choice $\oplus_3$, another prediction is made

\[
\begin{align*}
T & \vdash \!\! \vdash \!\! T \vdash \!\! \vdash \!\! L \\
1, T & \vdash \!\! \vdash \!\! T \vdash \!\! \vdash \!\! R \\
\oplus_3 \\
\end{align*}
\]

The reverse direction $G, \hat{C}, \hat{C} \vdash \!\! \vdash \!\! C \otimes \otimes \hat{G} \vdash \!\! \vdash \!\! C$ also holds.

\[
\begin{align*}
C, C^\perp & \vdash \!\! \vdash \!\! \perp \vdash \!\! \vdash \!\! L \\
C \oplus_3 T, C^\perp \oplus_3 1 & \vdash \!\! \vdash \!\! 1 \oplus_3 T \\
m, (C \oplus_3 T), C^\perp \oplus_3 1 & \vdash \!\! \vdash \!\! m \otimes (1 \oplus_3 T) \quad \otimes_L \\
CDB \otimes \otimes^2[m \otimes (C \oplus_3 T)], \otimes^2(C \oplus_3 1) & \vdash \!\! \vdash \!\! CDB \otimes \otimes^2[m \otimes (1 \oplus_3 T)] \\
CDB \otimes \otimes^2[m \otimes (C \oplus_3 T)], \otimes^2(C \oplus_3 1) & \vdash \!\! \vdash \!\! CDB \otimes \otimes^2[m \otimes (1 \oplus_3 T)] \\
G, C^\perp & \vdash \!\! \vdash \!\! \hat{G} \vdash \!\! \vdash \!\! C \\
G, \hat{C}, C^\perp & \vdash \!\! \vdash \!\! \hat{C} \otimes \otimes \hat{G} \vdash \!\! \vdash \!\! C \\
\otimes_L \\
\end{align*}
\]

or at the choice $\oplus_3$, another prediction is made

\[
\begin{align*}
\otimes^2(C \oplus_3 1) & \vdash \!\! \vdash \!\! CDB \otimes \otimes^2[m \otimes (1 \oplus_3 T)] \\
CDB \otimes \otimes^2[m \otimes (C \oplus_3 T)], \otimes^2(C \oplus_3 1) & \vdash \!\! \vdash \!\! CDB \otimes \otimes^2[m \otimes (1 \oplus_3 T)] \\
\hat{C} \vdash \!\! \vdash \!\! \hat{C} \\
\otimes_L \\
\end{align*}
\]
6.3 Dealing with Choices in \( \hat{A} \)

From the definition of \( \hat{A} \), \( \hat{A} \) is eventually \( A \) or 1 at some time point, after all the outcomes of choices are revealed. Whether \( A \) is retained or not from \( \hat{A} \) depends on the sequence of choice decisions which may be complex as the structure of \( \hat{A} \) can contain an arbitrary number of choices nested in time. To facilitate agent decision making on the outcome of \( \hat{A} \) (\( A \) is retained or not), it is convenient to compress all the choices in \( \hat{A} \) as a single choice to be made. In particular, the agent decides \( A \) to be retained (or not) in a single choice and follows the consequences of such decision. If choices are spread in time, this also corresponds to the strategy of deciding choices in advance.

Indeed, the notion of representative choice allows to compress all the choices in \( \hat{A} \) into one representative choice and specify as its constraints the sequence of decisions on these choices in \( \hat{A} \).

Consider the example of a formula being split up w.r.t. \( A \) as below

\[
G = O^a(A \oplus B) \&_2 O^b C
\]

\[
\hat{A} = O^a(A \oplus 1) \&_2 O^b 1.
\]

Eventually, \( \hat{A} \) can be either \( O^a A \) or \( O^b 1 \) or \( O^a 1 \) depending on the outcomes of choices. The sequence of decisions on choices to retain \( A \) is captured as

\[
(\frac{\hat{A}}{\hat{A}} \to L) \otimes O^a(\frac{\hat{A}}{\hat{A}} \to o L).
\]

A representative choice of \( A \) is expressed as \( O^a A \&_{r} O^b \), where \( y = b \) or \( y = a \) and determining conditions for \( \&_{r} \) are:

\[
CondL_{r} \vdash \frac{\hat{A}}{\hat{A}} \to L \otimes O^a(\frac{\hat{A}}{\hat{A}} \to o L)
\]

It can be seen that \( A \) is retained from \( \hat{A} \) if and only if the representative choice of \( A \) in \( \hat{A} \) is left chosen. Also, the outcome of \( \hat{A} \) after all the choices in \( \hat{A} \) are determined is also the outcome of this representative choice. Formally, we have the following result.

**Theorem 6.3.1.** Let \( \Gamma \) be a formula in the MCA fragment and \( \hat{A} \) be its split up w.r.t. \( A \) in \( \Gamma \). We have

\[
\hat{A} \vdash_{cc} \&_{cc} O^a A \&_{r} O^b 1
\]
where $\&_r$ is a representative choice of $A$ in $\Gamma$, $a$ indicates the corresponding time of the existence of $A$, and $b$ is an appropriate time depending on the outcomes of choices in $\hat{A}$.

This theorem can also be viewed as flattening the structure of $\hat{A}$.

**Proof:** we establish that $\&_r$ is also a representative choice of $A$ in $\hat{A}$ and given so, we prove the theorem by induction on the structure of $\hat{A}$ as described in Section A.1. □

Applying the above theorem to our running example, we obtain the results:

$$\hat{C} = C^2(C \oplus_3 1) \vdash_{cc} C^2 \&_{rc} C^2 1,$$

where $\text{Cond}_{L_{rc}} = \vdash C^2(\frac{\sqcup_1}{\sqcap_3} L)$.

$$\hat{T} = C^2(1 \oplus_3 T) \vdash_{cc} C^21 \&_{rl} C^2 T$$

where $\text{Cond}_{L_{rl}} = \vdash C^2(\frac{\sqcup_1}{\sqcap_3} L)$.

where $\&_{rc}, \&_{rl}$ are the representative choices of $C$ and $T$ in the goal of Peter respectively. Note that $\&_{rc}$ and $\&_{rl}$ are at the same time and have the same determining conditions, hence $\&_{rc}$ is the same as $\&_{rl}$.

Apply this result, Peter can turn his goal into concurrent sub-goals $CDB \otimes C^2 m \otimes (C^2 \oplus_4 C^2 1) \otimes (C^21 \oplus_4 C^2 2T)$, where the decision on $\oplus_4$ now is the same as that of $\oplus_3$ at the next two days. Therefore, agent Peter can achieve the two sub-goals $CDB \otimes C^2 m$ as discussed in Chapter 5 and sends the subgoal $(C^2 C \&_{rc} C^2 1)$ as a request to Ming and the subgoal $(C^2 1 \&_{rl} C^2 2T)$ as a request to Chaeng.

If Ming makes Chinese food, then $C^2 C \vdash_{[\&_{rc}] L}$ is derived. As the choice $\&_{rc}$ is left chosen, the other subgoal $(C^2 1 \&_{rc} C^2 2T)$ becomes $C^21$, which is also readily achievable. If Ming does not make Chinese food, there is a proof of $C^2 1$, where $[\&_{rc}] R$. This decision on the choice $\&_{rc}$ (choosing right) makes the subgoal $(C^2 1 \&_{rl} C^2 2T)$ becomes $C^2 2T$. Thus if all the subgoals are successful, this mechanism ensures that only one kind of food is made.

Hence, such splitting up of formulas allows Peter to concurrently and partially achieve his goal via different threads of interaction.
6.4 Reasoning about Splitting Up

In proof search based reasoning, reasoning is carried out by utilizing inference rules. To provide reasoning about splitting up goals or base commitments and resources or actions, we introduce inference rules that enable the splitting up. Specifically, the two theorems on the split ups

\[ \Gamma, \overrightarrow{A \perp} \vdash \overrightarrow{\Gamma - A} or \Gamma \vdash_{ce} \overrightarrow{A \perp} \rightarrow \overrightarrow{\Gamma - A} \]

and

\[ \overrightarrow{\Gamma - A} \otimes \overrightarrow{A} \vdash_{ce} \Gamma \]

give rise to inference rules on splitting up as below.

Let \( \overrightarrow{\Gamma - A} \) and \( \overrightarrow{A} \) denote the split ups of \( \Gamma \). Let \( \overrightarrow{A} \) and \( \overrightarrow{A \perp} \) denote the split ups of \( \Delta \). Let \( \overrightarrow{A} \) be a formula resulted from replacing \( A \) in \( \overrightarrow{A} \) by \( A \). In terms of reasoning (reading bottom up), such inference rules include the following.

Splitting up a goal or base commitment formula:

\[ \frac{\Gamma \vdash_{ce} \overrightarrow{\Delta - A} \otimes \overrightarrow{A} \Delta}{\Gamma \vdash_{ce} \overrightarrow{A}} \]

Combining split ups of a goal or base commitment formula:

\[ \frac{\Gamma \vdash_{ce} \Delta}{\Gamma \vdash_{ce} \overrightarrow{A} \overrightarrow{\Delta - A} } \]

\[ \frac{\Gamma \vdash_{ce} \Delta}{\Gamma \vdash_{ce} \overrightarrow{A} \overrightarrow{\Delta - A} } \]

Splitting up a resource or action formula:

\[ \frac{\Pi, \overrightarrow{A} \rightarrow \overrightarrow{\Gamma - A} \vdash_{ce} \Delta}{\Pi, \Gamma \vdash_{ce} \Delta} \]

\[ \frac{\Pi, \overrightarrow{\Gamma - A} \vdash_{ce} \Delta}{\Pi, \Gamma \vdash_{ce} \Delta} \]

Combining split ups of a resource or action formula:

\[ \frac{\Pi, \Gamma \vdash_{ce} \Delta}{\Pi, \overrightarrow{\Gamma - A}, \overrightarrow{\Delta} \vdash_{ce} \Delta} \]

Given that an appropriate implementation of these inference rules is in place, these rules equip agents with a proof search technique that extracts pairs of corresponding parts in the antecedent and succedent of a sequent. This can also be viewed as partial handling, which
is useful in a distributed and concurrent environment, as further discussed in Section 6.5. Moreover, because the theorems hold w.r.t. choice calculus, it is straightforward that these inference rules are sound in the choice calculus.

6.5 Distributed Concurrent Problem Solving

If we take the paradigm that problem solving is done via interaction among agents, then it is natural to explore distributed concurrent problem solving where goals or commitments and the resources or actions to fulfill them are distributed among concurrent interactions.

The theorems on splitting up formulas support such interactions at several levels.

Firstly, as mentioned, the theorems are applicable to splitting up an arbitrary sub-formula. Also, from their definitions, it is quite possible that the split up parts can be constructed without requiring prior specification on the number and forms of sub-parts but in an automated manner by agents. Hence, the division of a formula can be determined by agents in accordance with the situation. In other words, the division of formulas can be dynamic and an autonomous act by agents.

Secondly, partial and concurrent achievement of goals is enabled among interactions. Particularly, a goal (or a base commitment) $\Delta$ is applicable to an interaction where a part of it ($\hat{\Delta}$) can be fulfilled. $\Delta$ is then split up into $\hat{\Delta}$ and $\underline{\Delta} - \hat{\Delta}$ in which $\hat{\Delta}$ can be fulfilled by the interaction and hence removed and $\underline{\Delta} - \hat{\Delta}$ is what remains of the partially achieved goal $\Delta$. Similarly, a goal can be split up into several sub-parts ($\hat{\Delta}, \hat{\Gamma}, \hat{\Omega}...$) and these sub-parts can be concurrently explored in different interactions. Because the split ups are defined with dependencies among them, the effects of handling one part are reflected in the handling of others. When all sub-parts are concurrently or sequentially achieved, the original goal is fulfilled. This is formalized in the following theorem.

**Theorem 6.5.1. (Splitting up Goals or Base commitments)**

Let $\Delta$ be a goal or base commitment. Let $\hat{\Delta}_{\Delta}$ and $\underline{\Delta} - \hat{\Delta}$ be split ups of $\Delta$ w.r.t. a sub-formula $A$ in $\Delta$. We say $\Delta$ is fulfilled by resources or actions $\Gamma$ if $\Gamma \vdash \Delta$.

Then $\Delta$ is fulfilled if the two goals or base commitments $\hat{\Delta}_{\Delta}$ and $\underline{\Delta} - \hat{\Delta}$ are fulfilled and the fulfillment of $\Delta$ uses the same set of resources or actions.

**Proof:** based on the inference rules introduced in Section 6.4, the proof is straightforward.

Let $\Pi$ and $\Xi$ be resources/actions that fulfill $\hat{\Delta}_{\Delta}$ and $\underline{\Delta} - \hat{\Delta}$ respectively. In sequent form, we have $\Pi \vdash_{cc} \hat{\Delta}_{\Delta}$ and $\Xi \vdash_{cc} \underline{\Delta} - \hat{\Delta}$. Then
\[
\Pi \vdash_{cc} \Delta \quad \Xi \vdash_{cc} \Delta - A \otimes R
\]

which means that the resources/actions required to fulfill \( \Delta \) are \( \Pi \) and \( \Xi \). \( \square \)

Similarly, resources and actions can be partially and concurrently utilized among interactions. In an interaction, a resource or action \( \Gamma \) can be split up into \( \Gamma - A \) and \( \hat{A} \) of which \( \hat{A} \) can be utilized in an interaction of concern and subsequently removed. Split up parts can be utilized concurrently in different interactions with dependencies among them being captured and respected in their handling. The original resource or action \( \Gamma \) is (fully) utilized when all of its sub-parts are utilized. This also is formalized in the following theorem.

**Theorem 6.5.2. (Splitting up Resources or Actions)**

Let \( \Gamma \) be a resource or action. Let \( \hat{A}_\Gamma \) and \( \Gamma - A \) be split ups of \( \Gamma \) w.r.t. a sub-formula \( A \) in \( \Gamma \). Let \( \hat{A}_\Gamma^- \) be the result of replacing \( A \) in \( \hat{A}_\Gamma \) by \( A^- \). We say \( \Gamma \) fulfills \( \Delta \) if \( \Gamma \vdash \Delta \).

Then if the resources or actions \( [\hat{A}_\Gamma^-]+ \) and \( *\Gamma - A \) fulfill a combination of goals or base commitments then \( \Gamma \) also fulfills that combination.

**Proof:**

Due to the duality between goals or base commitments and resources or actions, the dual of \( \hat{A}_\Gamma^- \) is \( [\hat{A}_\Gamma^-]^+ \) and can be regarded as a resource or action. In fact, after having all the related choices decided and by applying De Morgan’s rules, \( [\hat{A}_\Gamma^-]^+ \) becomes either \( \bigodot^n A \) or \( \bigodot^m 1 \) for some \( n \), \( m \) and hence represents resources or actions.

Let \( \Xi \) and \( \Delta \) be goals or base commitments that are fulfilled by \( [\hat{A}_\Gamma^-]^+ \) and \( *\Gamma - A \) respectively. In sequent form, we have \( [\hat{A}_\Gamma^-]^+ \vdash_{cc} \Xi \) and \( *\Gamma - A \vdash_{cc} \Delta \). Then based on the inference rules introduced in Section 6.4, we have

\[
\begin{align*}
\Xi &\vdash_{cc} [A^-_\Gamma]^+ & \quad \vdash_{cc} R \\
\Xi &\vdash_{cc} A^-_\Gamma & \quad + L \\
\Xi &\vdash_{cc} \arrow{\hat{A}_\Gamma^-} & \quad *\Gamma - A \vdash_{cc} \Delta & \quad \arrow{L} \\
A^-_\Gamma &\vdash_{cc} *\Gamma - A, \Xi \vdash_{cc} \Delta & \quad \Gamma, \Xi \vdash_{cc} \Delta
\end{align*}
\]

which means that \( \Gamma \) can fulfill the goals/base commitments \( \Xi \) and \( \Delta \). \( \square \)
Summary

Given that goals or base commitments and resources or actions can be handled independently, these parts can be put in concurrent interactions. This means that the handling can be done in a distributed and concurrent manner. With a proper implementation, agents are enabled to explore each other’s resources or actions and achieve their goals dynamically and flexibly in a distributed manner among concurrent interactions.

As an application, we revisit our interaction model regarding how agents consider a proposal relevant as discussed in Section 4.3.7. Specifically, agents can apply the following strategy to deal with a proposal w.r.t. a goal or base commitment $\Delta$. The strategy explores partial achievement of the goal $\Delta$ with partial (or fully) utilization of the resources or actions provided by the proposal.

1. If the proposal can produce a formula $\Gamma$ which shares a sub-formula $A$ with $\Delta$ then $\Gamma$ is split into $\Gamma - A \otimes \hat{A}$ and $\Delta$ is turned into $\Delta - A \otimes \hat{A}'$.
   Applying $\hat{A} = \bigodot r_{rA} A \bigodot a$ and $\hat{A}' = \bigodot r_{rA'} A \bigodot b$.
   If
   \[
   \bigodot r_{rA} A \bigodot a \vdash_{cc} \bigodot r_{rA'} A \bigodot b
   \]
   is provable or can be inferred from
   \[
   \bigodot a \vdash_{cc} \bigodot b
   \]
   then accept the proposal, else reject the proposal.

2. If the proposal is accepted, the original goal or base commitment is replaced by $\Delta - A$.
   Fulfill conditions of the accepted proposal.

3. The accepted proposal is carried out and produces $\Delta - A \otimes \hat{A}$.
   Remove $\hat{A}$ and $\hat{A}'$.

$\Delta - A$ can be further fulfilled in other (possibly concurrent) interactions.

6.6 Summary

The chapter explored splitting up formulas with respect to an arbitrary sub-formula and defined dependencies among the split ups in the context of choices and resources. In a restricted logic fragment, we proved the equivalence between a formula and its split ups. The split up mechanism enables agents to partially and concurrently utilize resources or actions and achieve goals or base commitments among multiple interactions, which in turn is a fundamental step toward distributed concurrent problem solving.
Chapter 7

An Execution Model

Chapters 3 and 4 have introduced our framework for the specification of agent interaction protocols. We have demonstrated how protocols are specified and how agents might conduct reasoning based on protocol specifications and agent states. The next question is how to define an execution model for such specifications. In this chapter, we address various issues concerning an appropriate execution model.

In particular, Section 7.1 defines a fragment of TLL that is straightforward for execution purposes but still suitable for modeling in a finite time and resource context. The fragment is based on MCA (definition 9). Section 7.2 discusses a possible implementation by providing a mapping from the execution framework in Section 4.3 to pseudo-code provided in Appendix Chapter B. These describe in detail how agents interact based on acting to fulfill their goals and commitments. Section 7.3 discusses some further implementation issues.

7.1 A Fragment of TLL for Execution

Our modeling framework discussed so far has made use of a restricted form of first order temporal linear logic. In particular, the modeling of resources, actions, capabilities, and commitments uses the connectives and operators \( \otimes, \& , \oplus, \square, \diamond, \neg \) and the constants 1, \( \perp \). As noted previously, we use a purely propositional approach except for pre-commitments, in which universally quantified variables over agents are used for modeling location and ownership information. Consider the following example, where the merchant pre-commits that it will give a customer agent 2 junior cricket bats and a gift if that agent pays 20 dollars via Paypal or by credit card and provides an address.
\[20 \square \forall X, (20\$@X_X \odot (\text{via}_\text{PP}@M_X \oplus \text{by}_\text{cred}@M_X) \odot \text{addr}@X_X)\]
\[\rightarrow (20 \square \$@M_M \odot 2 \square \text{jr}_\text{cb}@X_X \odot 2\text{jr}_\text{cb}@M_M^\perp \odot \square \text{addr}@M_X \odot \square \text{gift}@X_X \odot \text{gift}@M_M^\perp)]\]

The variable \(X\) represents an agent where the respective resource is located or whom it belongs to. Being quantified universally over a domain of agents allows \(X\) to be assigned to any agent and hence allows the corresponding proposal to be applied to any agent of interest. Here, \(X\) is not an arbitrary object but must fall within the domain of agents and \(X\) only occurs at specific places (as the location and ownership information of resources or actions). Note that this is the only use of variables, i.e. in the preliminary step of finding an appropriate instance of a pre-commitment to be used. Once this is done, all reasoning is based on the propositional fragment. In particular, when a pre-commitment is proposed by one agent to another, all of its variables have already been instantiated and hence formulas of the pre-commitment and any corresponding conditional commitments are propositional.

Furthermore, as discussed in Section 6.1, it is necessary to restrict the logic fragment to ensure resource equivalence between a formula and its split ups. The resulting fragment is MCA which includes only the connectives \(\otimes, \&, \oplus, \odot\) and negation. Modeling of the notions of “anytime” and “sometime” was also revised in Section 6.1 to be based within this fragment as follows.

\[\square A \approx A \& (A \& (A \& \ldots \& A))\]
\[\diamond A \approx A \oplus (A \oplus (A \oplus \ldots \& A))\]

This is based on the intuition that all agent interactions will occur within time and resource limits. The upper limit on time \((T)\) is used where \(T\) refers to the farthest possible time point that the whole system could possibly reach. Under this assumption, the modeling of “anytime” and “sometime” can be thought as approximations of \(\square\) and \(\diamond\).

With these alternative modeling of “anytime”, “sometime”, the expressive power regarding non-determinism in time is still maintained. Indeed, the alternative modeling of \(\square\) allows agents to make an internal choice about the exact moment of a formula \(A\), which corresponds to outer non-determinism about time. Similarly, the alternative modeling of \(\diamond\) expresses an external choice about the precise moment of a formula \(A\), and so corresponds to inner non-determinism about time. Under this assumption of bounded time, we can reasonably assume
that the approximations preserve the expressiveness of non-determinism in time for $\Box$ and $\Diamond$.

The resulting fragment of TLL for modeling is then essentially MCA with a constraint on the modeling such that for any $\bigcirc^t \Gamma$, $0 \leq t \leq T$.

For convenience, in our modeling, we use $\Box$ and $\Diamond$ as abbreviations of their corresponding approximations described above.

The models of resources, actions, capabilities, goals, base commitments, and conditional commitments are then the same as described in Section 4.2.1 except that whenever formulas of $\Box A$ and $\Diamond A$ occur, they are replaced by their above approximations.

Following on from the discussion about message formats in Section 4.3.2, we arrive at the following formats for requests and proposals.

\[
\begin{align*}
\text{REQUEST } & \Delta \perp \\
\text{PROPOSE } & \bigcirc^n (\Gamma \rightarrow \Delta) \text{ or } \\
\text{PROPOSE } & \Box (\Gamma \rightarrow \Delta) \text{ or }
\end{align*}
\]

where $\Gamma$, $\Delta$ are formulas of the MCA fragment and the symbol $\Box$ is a shorthand for its approximation. Note that the proposed pre-commitment may be available at a specific time ($\bigcirc^n$) or any time ($\Box$).

The simplified fragment of TLL makes the complexity of proof search more manageable as approximations of $\Box$ and $\Diamond$ now require choices to be made (internally or externally) over a bounded range.

### 7.2 An Implementation Framework

As proposed in Chapters 3 and 4, the agents’ states are encoded with formulas representing resources, actions, capabilities, goals and commitments of the agents. Hence, in an implementation framework, each agent can be defined in terms of their resources, actions, rules (which refers to pre-commitments and capabilities), and goals or base commitments.

Protocols are then specified as a set of pre-commitments (also referred to as interaction rules) and base commitments at participating agents according to their roles in possible interactions. Hence, protocol specifications are also included in specifications of participating agents.

How agents interact by utilizing their own resources, actions and rules to fulfill their goals and commitments based on a protocol specification in our framework was described in
Section 4.3.

The activities of agents during interaction (as described in Section 4.3) can be organized into five groups as below. For each group of tasks, we also provide a detailed description for implementation in pseudo-code.

- **How the agent resolves a goal or base commitment**, as described in Section 4.3.6 and in pseudo-code in Section B.2.6:
  
  In order to achieve a goal or base commitment, an agent can act on its own or via interaction or a combination of both.
  
  If the agent acts on its own, its resources and actions can be used with its capabilities to derive the goal or base commitment via proof construction.
  
  If the agent cannot achieve a commitment or a goal by itself or if it so chooses, it will interact with other agents by making a request or a proposal.

- **How the agent makes a proposal**, as described in Section 4.3.4 and in pseudo-code in Section B.2.3:
  
  If the agent can find a pre-commitment whose resources or actions can fulfill a goal or commitment in a request, then the agent can propose that pre-commitment to the requesting agent. Also, if there is a pre-commitment whose resources or actions can fulfill a goal or commitment of the agent then the agent can propose the pre-commitment to another agent.

- **How the agent makes a request for a goal or base commitment**, as described in Section 4.3.3 and in pseudo-code in Section B.2.2.
  
  If the agent cannot find a relevant pre-commitment or if it so chooses, it will make a request for the commitment or goal to an appropriate agent.

- **How the agent responds to a request**, as described in Section B.2.4:
  
  Upon receiving a request, the requested agent searches for a pre-commitment of its own that is relevant to the request. If one is found, then the pre-commitment is proposed to the requesting agent. Otherwise, a failure notice will be returned to the requesting agent.

- **How the agent responds to a proposal**, as described in Section 4.3.4 and in pseudo-code in Section B.2.5:
  
  When a proposal is received, the recipient checks if there are some of its goals or base
commitments (including those in previous requests) that the respective pre-commitment is relevant to. If so, and the recipient cannot achieve these goals or base commitments by itself (or if it so chooses), the recipient will accept the proposal as described below. Otherwise, the recipient will send a message of rejection.

- **How the agent accepts a proposal:**
  If a proposal is accepted, the agent sends back a message of acceptance to the proposing agent. A conditional commitment is then formed by the proposing agent to the recipient agent from the pre-commitment of the proposal. If the expected outcomes of the proposed pre-commitment can only fulfill a part of a goal or base commitment of the recipient, then the goal or base commitment is split up to separate this part from the remainder, which is subject to further processing. As long as the recipient agent does not satisfy its requirements in the proposal (which are also treated as a form of goal or base commitment), the conditional commitment remains inactive. When the recipient fulfills the requirements as described in Section 4.3.5, the proposing agent will fulfill its specified commitments. If the outcomes of the conditional commitments are used partly then formulas of the outcomes can be split up to separate the parts to be used while the rest can be utilized elsewhere.

The requirements and commitments of the proposal can be treated as new goals or base commitments and hence fulfilling them may start another cycle and involve further interactions. Some further points of discussion about the execution framework are below.

Firstly, in our interaction model, we make use of the notion of *relevance* which is described in Section 4.3.7. If a pre-commitment is relevant to a goal or base commitment, then the pre-commitment can either be used internally, be proposed or be accepted (if it comes from another agent’s proposal). Checking relevance is based on checking if parts of the outcomes of the pre-commitment can logically derive some parts of a goal or base commitment, given appropriate choice strategies are applied. A strategy for considering a pre-commitment of a proposal is described in Section 6.5.

Secondly, the use of resource or action or the fulfillment of goal or base commitment can be either partial or total. Partial fulfillment (or use) is done by firstly separating the part to be fulfilled (or used) in the corresponding formula from the remainder via a split up, and then fulfilling (or using) the part. The theorems described in Section 6.5 on splitting up formulas of goals or base commitments and resources or actions enable the smooth integration of both partial and total handling of them.
Thirdly, for handling the fulfillment of goals or base commitments as a result of the provision of the necessary resources or actions, we deploy a mechanism based on modus ponens in TLL, where negative formulas are matched with the corresponding positive formulas to have the formulas of both removed. Removal of formulas means that the respective goals or base commitments are resolved or the respective resources or actions are consumed or carried out. In fact, the modus ponens $A \otimes (A \rightarrow \perp) \vdash \perp$ essentially provides a matching pair of the form $A, A^\uparrow \vdash \perp$ where $A^\uparrow$ represents the goals or base commitments which are dual to the resources and actions $A$ that are required to fulfill these goals or base commitments. Note that $\perp$ derives the empty succedent $(\perp \vdash)$, which in practice indicates that the removal of $A$ and $A^\uparrow$ is complete.

### 7.2.1 Agent Reasoning

In our TLL framework, as we consider what is provided ($\Gamma$) and what is required ($\Delta$) as two sides of a consequence relation in a sequent $\Gamma \vdash \Delta$, agent reasoning is then based on using inference rules to find a proof of the sequent. What is provided include resources, actions, capabilities and conditional commitments. What is required can be goals or base commitments. Successful reasoning means a successful proof search which begins with an initial sequent and steps through inference rules and ends with axiom sequents. Having a proof of a sequent means that agents have a successful plan in that given what is provided, the agent can obtain what is required by following the proof steps. Each proof step, which is an inference rule, can be translated to actions to be carried out by agents in terms of, for example, preparing resources, performing actions and capabilities, and making decisions on choices among resources and actions.

Apart from the standard inference rules of temporal linear logic sequent calculus, we provide the extra inference rules of choice calculus as described in Section 5.3 and for partial handling as described in Section 6.5 which together enable further agent reasoning. In summary, agent reasoning includes

- **Reasoning about partial handling of resources or actions and goals or base commitments**

  Agents can split up formulas of resources or actions and goals or base commitments and reason about using parts of them as discussed in Section 6.2 and Section 6.5.

  An example of reasoning steps that agents can carry out is as follows. Consider a sequent $\Pi, \Gamma \vdash \Delta$ where $\Gamma$ and $\Delta$ have a common basic TLL formula $\bigcirc^n A$, where $n$ is the time when $A$ exists. Denote $\Gamma \leftarrow \neg A$ and $\neg A_\Gamma$ as the two splits up of $\Gamma$. Denote $\Delta \leftarrow \neg A$
and $\hat{A}_\Delta$ as the two split ups of $\Delta$. The agent can reason about partial handling of $\Gamma$ and $\Delta$ by performing the proof search steps as follows.

$$
\begin{align*}
\hat{A}_\Gamma \vdash_{cc} \hat{A}_\Delta & \quad II, \Gamma \vdash \vdash_{cc} \Delta \vdash A \\
II, \Gamma \vdash A, \hat{A}_\Gamma \vdash_{cc} \Delta \vdash A \otimes \hat{A}_\Delta & \quad II, \Gamma \vdash A, \hat{A}_\Gamma \vdash_{cc} \Delta
\end{align*}
$$

The agent can then concurrently attempt to further reason about the two new sequents instead. Regarding the sequent

$$
\hat{A}_\Gamma \vdash_{cc} \hat{A}_\Delta
$$

given that $\hat{A}_\Gamma$ and $\hat{A}_\Delta$ are essentially $O^n A$ or $O^m 1$ for some $m, n$ (as a result of having all related choices determined), agents can reason about matching them up by applying the reasoning on the choices in them.

Regarding the sequent

$$
\Pi, \Gamma \vdash \vdash_{cc} \Delta \vdash A
$$

the same steps of reasoning about partial handling of them can also be subsequently applied on these parts of $\Gamma$ and $\Delta$.

- **Reasoning about choices**

Agents can reason about how various strategies with respect to internal choices and indeterminate possibilities (including changes in the environment) can be applied and their consequences and hence determine the right strategies. Discussions on these strategies and the corresponding inference rules are in Section 5.3.

In the example above, $\hat{A}_\Gamma$ and $\hat{A}_\Delta$ contain choices and their outcomes are either $O^n A$ or $O^m 1$ for some $m$. The agent can apply its strategies over these choices using the choice calculus sequent rules to prove the sequent

$$
\hat{A}_\Gamma \vdash_{cc} \hat{A}_\Delta
$$

or to infer it from the sequent

$$
O^n 1 \vdash_{cc} O^m 1
$$
which does not hold logically but makes sense in the context of resources as $\bigcirc^n 1$ means having no resource and just acts as a unit of the connective $\bigotimes$. In fact, the sequent corresponds to the case $A$ is not retained in $\Gamma$ and $\Delta$ and hence does not exist. Correspondingly, $\hat{A}_\Gamma$ and $\hat{A}_\Delta$ will become $\bigcirc^n 1$ and $\bigcirc^m 1$ (for some $n, m \geq 0$) which means no resource or action involved. Therefore, the strategy for agents in this case should be ignoring the parts $\hat{A}_\Gamma$ and $\hat{A}_\Delta$ and continue with the sequent

$$\Pi, \Gamma \vdash A \vdash_{cc} \Delta \vdash A$$

- **Reasoning about Pre-commitments with respect to Achieving Commitments**

A combination of reasoning about partial handling and reasoning about choices like in the above example enables agents to reason about how a pre-commitment is relevant to a goal or base commitment. By reasoning about partial handling, the agent can bring up the pairs that contain a part from outcomes of the pre-commitment and a part from the goal or base commitment for consideration. By reasoning about various strategies on relevant choices in these parts, the agent can determine whether proof can be found relating these pairs (such as a proof of the sequent $\hat{A}_\Gamma \vdash_{cc} \hat{A}_\Delta$) and consequently if the pre-commitment is relevant.

Being able to reason about the relevance of a pre-commitment to a goal or base commitment further enables agents to reason about making proposals to other agents or accepting proposals from them. Hence, this reasoning ability promotes agent autonomy and makes interaction more flexible.

In addition, agent reasoning can also be based on customized rules which are those deducible from the TLL sequent calculus rules and aimed at helping agents to find shortcuts in proof construction especially in the resource context. These rules are ($n \geq 0$):

$$\Gamma \vdash \bigcirc^n A \quad \Gamma, \bigcirc^n A \vdash \Delta \quad \Gamma \vdash \bigcirc^n A \otimes \bigcirc^n B \quad \Gamma \vdash \bigcirc^n (A \otimes B)$$

$$\Gamma \vdash \bigcirc^n A \otimes \bigcirc^n B \quad \Gamma \vdash \bigcirc^n (A \otimes B)$$

$$\Gamma \vdash \Box A \otimes \Box B \quad \Gamma \vdash \Box A \otimes \Box B \quad \Gamma \vdash \Box (A \otimes B) \quad \Gamma \vdash \Box A \otimes \Box B \quad \Gamma \vdash \Box (A \otimes B)$$
Further Implementation Issues

Note that in the reasoning process, agents can plan for an application of a pre-commitment of the form $\Box^n(\Gamma \vdash \Delta)$ at any time between now and $\Box^n$. Particularly, we can perform the linear implication now $\Box^n \Gamma \otimes \Box^n(\Gamma \rightarrow \Delta) \Rightarrow \Box^n \Delta$, or at $\Box^n$ like $\Box^n[\Gamma \otimes (\Gamma \rightarrow \Delta)] \Rightarrow \Box^n \Delta$ or at some time in between $\Box^m$ like $\Box^m[\Box^{n-m} \Gamma \otimes \Box^{n-m}(\Gamma \rightarrow \Delta)] \Rightarrow \Box^n \Delta$. In fact, the exact time is quite flexible, depending on when the requirements of the linear implication $\Box^n \Gamma$ becomes available to the agent.

7.3 Further Implementation Issues

We discuss further issues regarding implementation of such an execution framework below.

7.3.1 Proof Search in TLL

Our execution framework is based on proof search in the MCA fragment of TLL. We have provided extra sequent rules in the choice calculus and partial handling as a basis for proof search techniques in our execution framework. Given the standard sequent rules and our extra rules in the fragment MCA, an implementation of such sequent rules is a straightforward implementation of existing theorem proving techniques and we believe the work can be carried out by a skilled programmer. Hence, our pseudo-code description of an execution framework is based on the assumption that the implementation of the appropriate proof search techniques is unproblematic.

7.3.2 Mapping Different Agents’ Time Systems

Time in TLL is discrete and measured against a system of time points with reference to the time point “now”. Time points in TLL indeed have no absolute values but relative values with respect to now. Given an open system like a multi-agent system, each agent might have a different period between one time point to the next in its time system. This discrepancy is likely to cause difficulties as three time points with a short period may be shorter than two time points with a long period. Therefore, this issue must be resolved in an implementation.

One approach to ensure that a time point means the same across agents’ systems is to make sure that the period from one time point to the next is fixed and constant among these time systems. This period can be regarded as a time unit. Hence, every reference to the time point “now” can be measured by the number of time points multiplied by the time unit and has the same value in all systems. Moreover, this requirement of a constant period between
consecutive time points can be placed at the design phase of the agent systems so that every formula will be assigned with the correct time points.

Another approach is to map all agents’ time systems into the same time system so that any time reference is applicable and interpreted the same to all agents. In particular, time points of formulas of each agent are compared by their actual or absolute values and all mapped into one unified time system. This process bring formulas of agents to their correct time points.

For example, let agent $\alpha$ have formulas $A$, $\bigcirc B$ and $\bigcirc \bigcirc C$ while agent $\beta$ has formulas $\bigcirc D$, and $\bigcirc \bigcirc \bigcirc E$. Denoting the layout in time as follows:

In agent $\alpha$’s time system: $0_\alpha(A)$, $1_\alpha(B)$, $2_\alpha(C)$

In agent $\beta$’s time system: $0_\beta$, $1_\beta(D)$, $2_\beta$, $3_\beta(E)$

Such a time synthesis mechanism will collapse the two relative time systems into one system. By comparing their respective actual time, we can detect their relative order, for example, $0_\alpha$ is in between $1_\beta$, and $2_\beta$: $1_\beta,0_\alpha,2_\beta$; $2_\beta,1_\alpha,3_\beta$; and $2_\alpha$ is after $3_\beta$: $3_\beta,2_\alpha$, the rearrangement will become:

$$0_{\alpha \beta} 1_{\alpha \beta}(D) 2_{\alpha \beta}(A) 3_{\alpha \beta} 4_{\alpha \beta}(B) 5_{\alpha \beta}(E) 6_{\alpha \beta}(C)$$

The formulas of agents $\alpha$ and $\beta$ are then adjusted as

Agent $\alpha$: $\bigcirc^2 A$, $\bigcirc^4 B$ and $\bigcirc^6 C$

Agent $\beta$: $\bigcirc D$, and $\bigcirc^5 E$.

Moreover, there are situations where $\bigcirc$ is locally interpreted, i.e. with respect to a local reference time point. Formulas nested inside a bracket are typically of this type. An example is $\bigcirc^2(A \otimes \bigcirc B)$, where $\bigcirc B$ means local within the bracket and hence the absolute time should be $\bigcirc^3 B$. In these cases, the actual time points is obtained by adding the local relative time to the actual time of the local reference time point.

It can be seen that the first approach does not require synchronization of time and adjustment of formulas of agents’ states but has to enforce the requirement of having constant period between consecutive time points. On the other hand, the second approach allows the freedom of designing formulas with a suitable local time system at the cost of synchronization for the whole system every time a new local time system is introduced.

Note that operating in different time zones does not cause any issue in TLL time systems because all the time points are referenced to the time point “now” such as “two time points from now”, “three time points from now”, etc. As now means the present time universally,
Further Implementation Issues

176

having the same reference point means that these time points can be interpreted in the same way.

For the purpose of focusing on the interaction among agents, we assume that either one of the above mentioned approaches must be in place so that time references among agents’ systems are correctly interpreted.

7.3.3 Derivations among Conjunctions and Disjunctions in Resource Contexts

Further observations about deriving formulas in a resource context reveals some useful results.

Although $\circ^n A \otimes \circ^p B \neq \circ^n A \& \circ^p B$, in terms of resources, having concurrently two resources $\circ^n A$ and $\circ^p B$ will enable agents to derive anyone of $\circ^n A$ and $\circ^p B$ and hence can cope with an indeterminate possibility between them. Specifically, given $\circ^n A \& \circ^p B \vdash \circ^n A \& \circ^p B$

, in order to make the derivation, the owner agent needs an ability to choose either $\circ^n A$ or $\circ^p B$ to cope with any possibility of the requirement $\circ^n A \& \circ^p B$ being $\circ^n A$ or being $\circ^p B$. A necessary condition to have such ability is having both $\circ^n A$ and $\circ^p B$, or $\circ^n A \otimes \circ^p B$ available.

As a result, in terms of resource derivations, from $\circ^m(\circ^n A \otimes \circ^p B)$, agents can derive $\circ^m(\circ^n A \& \circ^p B)$. We denote $\vdash_r$ as the resource derivation relationship then $\circ^m(\circ^n A \otimes \circ^p B) \vdash_r \circ^m(\circ^n A \& \circ^p B)$. Certainly, logical derivation implies resource derivation but the reverse is not necessarily true.

We then have the following:

$\circ^{m+n} A \otimes \circ^{m+p} B \vdash \circ^m(\circ^n A \otimes \circ^p B) \vdash_r \circ^m(\circ^n A \& \circ^p B) \vdash \circ^{m+n} A \& \circ^{m+p} B \vdash \circ^{m+n} A \oplus \circ^{m+p} B \vdash \circ^m(\circ^n A \oplus \circ^p B)$

Therefore, in order to derive a conjunction or disjunction, we can derive any of the precedents in this order of derivation.

Moreover, according to the order of derivation, as a shortcut strategy, agents can try to derive $\circ^{m+n} A \otimes \circ^{m+p} B$

in order to meet any requirement of
Further Implementation Issues

\[ \circ^m (\circ^n A \otimes \circ^p B) \text{ or } \circ^m (\circ^n A \& \circ^p B) \text{ or } \circ^{m+n} A \& \circ^{m+p} B \]

and try to derive

\[ \circ^{m+n} A \& \circ^{m+p} B \]

to meet any requirement of:

\[ \circ^{m+n} A \oplus \circ^{m+p} B \text{ or } \circ^m (\circ^n A \oplus \circ^p B) \]

One major benefit of this shortcut strategy is that to deal with any conjunctions or disjunctions with reference to the future time, the agent can prepare a corresponding present multiplicative conjunction and hence can attempt to prepare each conjunct individually. Each conjunct can be subsequently dealt with in the same manner so that in the end, the agent has to prepare a set of basic TLL formulas of the form \( \circ^i A, i \geq 0 \), where \( A \) is a LL proposition. Consequently, dealing with an arbitrary goals or base commitments formula can be reduced to preparing all the corresponding basic TLL formulas of the goals or base commitments.

The cost of this approach is the extra reservation of resources which comes directly from derivations of the form

\[ \circ^{m+n} A \otimes \circ^{m+p} A \vdash_r \circ^m (\circ^n A \& \circ^p B) \]

which prepares both \( \circ^{m+n} A \) and \( \circ^{m+p} A \) while only one of them will be required. In fact, this can be regarded as a safe approach as it requires to prepare all possibilities.

For example, consider a requirement of

\[ \circ (\circ A \& \circ^2 B) \otimes (C \oplus \circ^3 D) \]

for a given current state of an agent

\[ ( \& \sum_{i=0}^{T-1} \circ^i A ) \otimes \circ^3 B \otimes C. \]

Using this approach, the requirement can be turned into another requirement

\[ \circ^2 A \otimes \circ^3 B \otimes C. \]
Hence, matching with the resources available at the current state can be straightforward, given
\[ \land_i \land_{i=0}^{T-1} \land_i \land_i A \land_2 A. \]

If this approach fails then the agent can attempt as normal via proof search.

This shortcut approach to derivations of conjunctions and disjunctions can be implemented as a heuristic which is supplementary to the standard implementation of proof construction in the MCA fragment. In many cases, this approach can provide simple and elegant solution for agents when dealing with resources.

7.3.4 Other issues

There are other important issues for an implementation to address. Such issues, for instance, include handling concurrency, deadlocks, and processing message formats. However, these issues are typical in most multi-agent systems and not peculiar to our execution framework. Hence, our description of an execution framework is assumed to be combined with existing mechanisms that handle these issues.

7.4 Summary

The chapter addressed implementation issues for the execution framework described in Section 4.3. A simple but effective fragment of TLL for execution purposes which contains only the connectives and operators \( \otimes, \land, \lor, \land \) and negation was discussed. On the one hand, the fragment allows distributed and concurrent handling of goals or base commitments and resources or actions based on the split up mechanism discussed in Chapter 6. On the other hand, the fragment is shown to retain enough expressive power for our modeling tasks.

Moreover, various issues for the execution framework based on proof search were discussed. In particular, we discussed how proof search can be used to provide agents’ reasoning about the relevance of a resource or action or a pre-commitment with respect to a goal or base commitment, and to provide handling of fulfillment of a goal or base commitment given a provision of necessary resources and actions. We also discussed how a request is formed and responded to as well as how a proposal is formed and accepted. Underlying these issues are the strategies of agents in dealing with choices.
Chapter 8

Examples and Discussion

In this chapter, we discuss the strengths and weaknesses of modeling interaction in temporal linear logic (TLL) and the choice of execution framework for TLL specifications. Moreover, we discuss how our approach helps to obtain flexibility in agent interaction.

Section 8.1 introduces a detailed example of the modeling, specification and highlights of execution of interaction among agents. Another example is provided in Section B.4. These examples provide a concrete basis for the reflective discussions in Section 8.2 on modeling and in Section 8.3 on flexibility in agent interaction. Moreover, we examine an alternative approach to the execution framework that is based on timed Petri nets in Section 8.4 and discuss the differences with our choice of using proof search in TLL for executing specifications. Timed Petri nets (TPNs) are visual and mathematical tools for modeling concurrent systems in time. By using TPNs as an execution model, we can take advantage of various existing tools and techniques for TPNs to address issues of execution and verification of TLL specifications.

8.1 Concert Ticket Example

To facilitate discussions on our TLL modeling and execution framework for agent interaction, we will make use of the following example. Comparisons with traditional approaches to specifying protocols are also provided as part of the evaluation.

Description

Three agents, (a musician, a writer and an artist) try to exchange resources so that they can all satisfy their goals.
The musician agent has a goal of obtaining two tickets to the concert at the second next time point. It is willing to sell its music in mp3 format for 5 dollars per hour of music. It has 15 dollars, and also has a book that can be exchanged for 25 dollars.

The writer agent does not have any money but has a ticket, which is available from now until the third next time point, a CD player, and an mp3 player. The writer can sell the ticket for 20 dollars and the mp3 player for 15 dollars. The writer likes to read books and listen to music and hence has a goal of doing these at the second next time point. Listening to music requires two hours of mp3 music or equivalent. The writer can choose to listen on the mp3 player or the CD player. It also has the capability to convert two hours of mp3 music into a CD.

The artist agent intends to give a performance at the third next time point. To do so, it requires an mp3 player, and two hours of mp3 music. The artist only has a ticket which is also available from now until the third next time point and can be exchanged for 25 dollars.

One solution for the agents to interact to achieve their goals is as follows.

**Sale of a ticket from the writer to the musician**
- The musician makes a request for a ticket at the second next time point to the writer.
- The writer responds by proposing a sale of a ticket available from now until the third next time point for 20 dollars.
- The musician accepts, commits to paying 20 dollars and gets the ticket.

**Sale of a ticket from the artist to the musician**
- The musician makes a request for a ticket at the second next time point to the artist.
- The artist proposes the sale of a ticket available from now until the third next time point for 25 dollars.
- The musician accepts, commits to paying 25 dollars and gets the ticket.

**Sale of a book from the musician to the writer**
- The writer makes a request for a book to the musician.
- The musician proposes selling the book for 25 dollars.
- The writer accepts, takes the book and pays 5 dollars. The commitment of the musician to pay 20 dollars to the writer is removed.

**Sale of an mp3 player from the writer to the artist**
- The artist makes a request for an mp3 player to the writer.
- The writer proposes selling the mp3 player for 15 dollars.
- The artist accepts, gets the mp3 player and pays 15 dollars to the writer.

**Sale of mp3 music from the musician to the writer**
- The writer makes a request for 2 hours of mp3 music to the musician.
- The musician proposes selling each hour of mp3 music for 5 dollars.
- The writer accepts, obtains the music and pays 10 dollars to the musician.

**Sale of mp3 music from the musician to the artist**
- The artist makes a request for 2 hours of mp3 music to the musician.
- The musician proposes selling each hour of mp3 music for 5 dollars.
- The artist accepts, obtains the music and pays 10 dollars to the musician.

Hence, from the interaction, the artist gets 25 dollars from selling its ticket, pays 10 dollars to the musician for 2 hours of mp3 music and 15 dollars to the writer for the mp3 player. It does not have any more money but gets an mp3 player and 2 hours of mp3 music for its performance. Hence, the artist achieves its goal. The writer gets 20 dollars for selling its ticket, 15 dollars for selling the mp3 player, pays the musician 25 dollars for a book, 10 dollars for 2 hours of mp3 music and hence gets no debt. The writer then has a book for reading and music for listening and therefore achieves its goal. The musician pays 45 dollars for two tickets, gets 20 dollars for selling mp3 music, 25 dollars for selling its book and hence, retains its 15 dollars. The musician also achieves its goal.

It can be observed that the requirements for each sale are the item(s) for sale and a payment amount. Given the item(s) for sale can be made available later by the sale agent and the payment amount can be regarded as a debt, the sale can start anytime during the interaction. In other words, these sales are independent and are not subject to any constraint on the order of execution. Hence, interaction among the agents can vary in the order of sales of tickets, book, mp3 player and mp3 music while still achieving all agents’ goals. Also, some of the sales may be initiated by two options, a request or a proposal to sell goods (to gain money for the commitments to pay). Such flexibilities should be captured in the specification of the interaction.

### 8.1.1 Our Approach to Specifying Concert Ticket Interaction

A specification of the interaction in our approach is detailed below. In the specification, pre-commitments are also referred to as rules for interaction.

**Musician**
The musician has a goal of obtaining two tickets at the second next time point, which is one
time point before the concert and has a book and 15 dollars, and unlimited access to mp3 songs:

\[
[\bigcirc^2 \text{ticket} @ M_M] \cong \bigotimes \bigcirc^2 \text{ticket} @ M_M \otimes \eta \Box mp3 @ M_M \otimes \square \text{book} @ M_M \otimes 15 \square @ M_M
\]

The musician can offer to anyone access to her mp3 music at 5 dollars per hour:

Rule 1 - \( \eta \Box [\forall X, 5\$ @ X_X \to \Box mp3 @ X_X \otimes mp3 @ M_M \otimes 5 \square @ M_M] \)

The musician can also offer her book for 25 dollars:

Rule 2 - \( \Box [\forall X, 25\$ @ X_X \to \Box \text{book} @ X_M \otimes book @ M_M \otimes 25 \square @ M_M] \)

**Writer**

The writer has a goal of listening to music and reading a book at the third next time point. The writer has a ticket available from now until the next third next time point, a CD player and an mp3 player:

\[
[\bigcirc^2 \text{music} @ W_w] \cong \bigotimes \bigcirc^2 \text{book} @ W_w \otimes \bigwedge \_i \geq 0 (\bigcirc' \text{ticket} @ W_w) \otimes \Box CD \_\text{player} @ W_w \otimes \Box mp3 \_\text{player} @ W_w
\]

The writer can listen to music directly from either a CD music or 2 hours of mp3 music:

Rule 3 - \( \Box [(CD @ W_w \otimes CD \_\text{player} @ W_w ) \oplus (2mp3 @ W_w \otimes mp3 \_\text{player} @ W_w ) \to music @ W_w ] \)

The writer can write a CD from 2 hours of mp3 music:

Rule 4 - \( \eta \Box [2mp3 @ W_w \to CD @ W_w ] \)

The writer can sell his ticket for 20 dollars

Rule 5 - \( \Box [\forall X, 20\$ @ X_X \to \text{ticket} @ X_X \otimes \text{ticket} @ W_w \cong 20 \square @ W_w ] \)

The writer can also offer his mp3 player for 15 dollars:

Rule 6 - \( \Box [\forall X, 15\$ @ X_X \to mp3 \_\text{player} @ X_X \otimes mp3 \_\text{player} @ W_w \cong 15 \square @ W_w ] \)

**Artist**

The artist has a goal of a performance at the second next time point and has a ticket to the concert which is available from now until the third next time point:

\[
[\bigcirc^3 \text{perform} @ A_A] \cong \bigotimes \bigwedge \_i \geq 0 (\bigcirc' \text{ticket} @ A_A )
\]

The artist can sell the ticket for 25 dollars:

Rule 7 - \( \Box [\forall X, 25\$ @ X_X \to \text{ticket} @ X_X \otimes \text{ticket} @ A_A \cong 25 \square @ A_A ] \)

The artist needs 2 hours of mp3 music and an mp3 player to perform

Rule 8 - \( \Box [2mp3 @ A_A \otimes mp3 \_\text{player} @ A_A \to \text{perform} @ A_A] \)
At the end of the interaction, a successful protocol run must satisfy the criteria: all the commitments

\[
[\Box^2 \text{ticket}\mathbin{\ominus} M_A] \land, [\Box^2 \text{ticket}\mathbin{\ominus} M_A] \land, [\Box^2 \text{music}@W_w] \land, [\Box^2 \text{book}@W_w] \land \text{ and } [\Box^3 \text{perform}\mathbin{\ominus} A_A] \land
\]

are fulfilled and there are no pending commitments resulting from applications of the rules during interaction.

A summary of all the rules is provided in Table 8.1. Note that the set of all the rules (pre-commitments) also forms a specification of the protocol for interaction scenarios of the example.

<table>
<thead>
<tr>
<th>Musician</th>
<th>Rule 1 ( \eta \Box [\forall X, 5\circ X_x \rightarrow \square \text{mp3}@X_x \otimes \text{mp3}@M_M^X \otimes 5 \circ \Box @ M_M] )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rule 2</td>
<td>( \Box [\forall X, 25\circ X_x \rightarrow \square \text{book}@X_x \otimes \text{book}@M_M^X \otimes 25 \circ \Box @ M_M] )</td>
</tr>
<tr>
<td>Writer</td>
<td>Rule 3 ( \Box [\square \text{CD}@W_w \otimes \text{CD}<em>\text{player}@W_w) \otimes (2\text{mp3}@W_w \otimes \text{mp3}</em>\text{player}@W_w) \rightarrow \text{music}@W_w )</td>
</tr>
<tr>
<td>Rule 4</td>
<td>( \eta \Box [\square \text{mp3}@W_w \rightarrow \text{CD}@W_w] )</td>
</tr>
<tr>
<td>Rule 5</td>
<td>( \square [\forall X, 20\circ X_x \rightarrow \text{ticket}@X_x \otimes \text{ticket}@W_w^X \otimes 20 \circ \Box @ W_w] )</td>
</tr>
<tr>
<td>Rule 6</td>
<td>( \square [\forall X, 15\circ X_x \rightarrow \square \text{mp3}<em>\text{player}@X_x \otimes \text{mp3}</em>\text{player}@W_w^X \otimes 15 \circ \Box @ W_w] )</td>
</tr>
<tr>
<td>Artist</td>
<td>Rule 7 ( \square [\forall X, 25\circ X_x \rightarrow \text{ticket}@X_x \otimes \text{ticket}@A_A^X \otimes 25 \circ \Box @ A_A] )</td>
</tr>
<tr>
<td>Rule 8</td>
<td>( \square [\square \text{mp3}@A_A \otimes \text{mp3}_\text{player}@A_A \rightarrow \text{perform}@A_A] )</td>
</tr>
</tbody>
</table>

Table 8.1: Summary of Specification Rules for Concert Ticket Example

### 8.1.2 Construction of Concert Ticket Interaction

Given this specification, this section demonstrates how interactions are constructed based on agents’ reasoning about their commitments, states, and which rules to use to fulfill them. A model for guiding reasoning about rules and for negotiation among agents was described in Section 4.3.1.

To simplify the proof, we make use of some shortcut inference rules such as \( [\Box \square] \) and \([\text{app}]\), which are described in Section 4.3.9. Note that \([n]\) in the title of a rule denotes multiple applications of that rule. We also make use of the splitting mechanism of a compound formula described in Section 6.2. For example, a conjunct \( \Box^n A \) can be extracted from a multiplicative
conjunction $\bigcirc^n(A \otimes B \otimes C)$. For simplicity of discussion, we only mention the desired split ups without showing details of the corresponding proof search steps.

1. Initial States

Because none of the artist, musician and writer can achieve their goals by themselves, they will make a request for their goals to the other agents.

Musician:
The positive formula of $[\bigcirc^2 \text{ticket}@M]_M$ is $\bigcirc^2 \text{ticket}@M$ which the musician cannot derive by itself. To fulfill the goal of having two tickets, the musician can make requests of the goal formula to other agents. This is discussed further in step 2.

M to W: REQUEST $[\bigcirc^2 \text{ticket}@M]_M$ and

M to A: REQUEST $[\bigcirc^2 \text{ticket}@M]_M$

Writer:
The goal $[\bigcirc^2 \text{music}@W]_W$ can be fulfilled by deriving the corresponding resource $\bigcirc^2 \text{music}@W$, using an application of Rule 3 as follows. Denoting Rule 3 as $\square[L3 \rightarrow R3]$, we then have

$$\vdash \bigcirc^2 L3 \quad \square[L3 \rightarrow R3] \vdash \bigcirc^2 R3 \quad \text{app}$$

where $\bigcirc^2 R3 = \bigcirc^2 \text{music}@W$.

We abbreviate $\text{mp3}@W$ as $\text{mpw}$, $\text{CD}@W$ as $\text{cpw}$, $\text{mp3_player}@W$ as $\text{mpw}$, and $\text{music}@W$ as $\text{muw}$.

The conditions for the application of Rule 3 are

$$\bigcirc^2 L3 = \bigcirc^2 [(\text{cpw} \otimes R) \oplus (\text{mpw} \otimes \text{mpw})]$$

which are further desired. The conditions can be analyzed into sub-conditions as follows.

$$\vdash \bigcirc^2 \text{cd} \quad \vdash \bigcirc^2 \text{cpw} \quad \text{or} \quad \vdash \bigcirc^2 \text{mpw} \quad \vdash \bigcirc^2 \text{cpw}$$

The writer then makes these sub-conditions as new subgoals, i.e. $([\bigcirc^2 \text{mp3}@W]_W^\perp$ and $[\bigcirc^2 \text{mp3_player}@W]_W^\perp$) or $([\bigcirc^2 \text{CD}@W]_W^\perp$ and $[\bigcirc^2 \text{CD_player}@W]_W^\perp$) and the writer can make the choice.

\(\text{--(1)--}\)
Regarding the former choice, the writer has
\[\Box \text{mp3\_player}@W_w \vdash \bigodot^2 \text{mp3\_player}@W_w\]

but cannot derive \(\bigodot^2 \text{mp3}@W_w\) by itself.

Regarding the latter, \(\Box \text{Cd\_player}@W_w \vdash \bigodot^2 \text{Cd\_player}@W_w\).
\(\bigodot^2 \text{Cd}@W_w\) can be derived by Rule 4 as follows. Denote Rule 4 as \(\square [L4 \rightarrow R4]\) then
\[
\frac{\bigodot^2 L4}{\square [L4 \rightarrow R4] \vdash \bigodot^2 R4} \text{ app}
\]

where \(\bigodot^2 R4 = \bigodot^2 \text{Cd}@W_w\). The conditions for the application of Rule 4 are \(\bigodot^2 L4 = \bigodot^2 \text{mp3}@W_w\).

Therefore, either choice requires \(\bigodot^2 \text{mp3}@W_w\), which cannot be achieved by the writer alone.

In order to derive \(\bigodot^2 \text{mp3}@W_w\), the writer can possibly make a request for \([\bigodot^2 \text{mp3}@W_w] \perp\) to other agents. This is further discussed in step 6.

W to A: REQUEST \([\bigodot^2 \text{mp3}@W_w] \perp\) and
W to M: REQUEST \([\bigodot^2 \text{mp3}@W_w] \perp\)

The goal \([\bigodot^2 \text{book}@W_w] \perp\) cannot be derived by the writer alone. Consequently, he can possibly make requests of the book to other agents. This is discussed further in step 4.

W to A: REQUEST \([\bigodot^2 \text{book}@W_w] \perp\) and
W to M: REQUEST \([\bigodot^2 \text{book}@W_w] \perp\)

**Artist:**

An application of Rule 8 can help to achieve its goal of \(\bigodot^3 \text{perform}@A_A\) as follows. Denoting Rule 8 as \(\square [L8 \rightarrow R8]\), we then have
\[
\vdash \bigodot^3 L8 \frac{\square [L8 \rightarrow R8] \vdash \bigodot^3 R8}{\text{app}}
\]

where \(\bigodot^3 R8 = \bigodot^3 \text{perform}@A_A\).

The conditions for the application of Rule 8 are \(\bigodot^3 L8 = \bigodot^3 [\text{mp3}@A_A \otimes \text{mp3\_player}@A_A]\). The conditions can be analyzed into sub-conditions as below.
Taking these sub-conditions as sub-goals means that the artist will look for subgoals of \([\blacksquare^{3 \text{mp}\_\text{player} @ A}] \perp\) and \([\blacksquare^{3 \text{mp}3 @ A}] \perp\). As the artist cannot achieve these sub-goals by itself, it can make corresponding requests to other agents. This is further discussed in steps 5 and 7.

A to W: REQUEST \([\blacksquare^{3 \text{mp}\_\text{player} @ A}] \perp\) and
A to W: REQUEST \([\blacksquare^{3 \text{mp}3 @ A}] \perp\) and
A to M: REQUEST \([\blacksquare^{3 \text{mp}\_\text{player} @ A}] \perp\) and
A to M: REQUEST \([\blacksquare^{3 \text{mp}3 @ A}] \perp\)

Given the reasoning of the agents as above, the actual interaction can start with any of the requests above. Some of these requests will fail, because the requested agents do not have the corresponding resources available, as in the cases below.

W to A: REQUEST \([\blacksquare^{2 \text{mp}3 @ W}] \perp\)
A to W: REQUEST \([\blacksquare^{3 \text{mp}3 @ A}] \perp\)
A to M: REQUEST \([\blacksquare^{3 \text{mp}\_\text{player} @ A}] \perp\)
W to A: REQUEST \([\blacksquare^{2 \text{book} @ W}] \perp\)

The other requests will involve the agents in more interaction. We present all the sections of sales below. Note that, the numbering only identifies the sections and does not imply the sequence of them.

**2. Sale of a ticket from the writer to the musician**

The sale can start by a request from the musician or by a proposal from the writer to sell its ticket in order to gain some money if the writer has some commitments of deriving money. Either way, the interactions share a common part, starting from the proposal by the writer of Rule 5.

The musician makes a request for a ticket to the concert which clearly specifies that the ticket is desired at the second next time point.

M to W: REQUEST \([\blacksquare^{2 \text{ticket} @ M}] \perp\)

To fulfill \([\blacksquare^{2 \text{ticket} @ M}] \perp\), a proof can be constructed from Rule 5 of the writer to derive \(\blacksquare^{2 \text{ticket} @ M}\).
Consider an application of Rule 5 as follows. Denoting Rule 5 (with the assignment $X = M$) as $\Box[L5 \rightarrow R5]$, we then have
\[
\vdash \Box^2 L5 \quad \vdash \Box^2 R5 \quad app
\]
where $\Box^2 R5 = \Box^2 [ticket @ M_M \otimes ticket @ W_w \uparrow \otimes 20 \Box @ W_w]$, which can be split up to make the sub-part $\Box^2 [ticket @ M_M \otimes ticket @ W_w \uparrow \otimes 20 \Box @ W_w]$ available to fulfill $[\Box^2 [ticket @ M_M \otimes ticket @ W_w \uparrow \otimes 20 \Box @ W_w]]$. In other words, the outcomes of an application of Rule 5 can be partially used to achieve $[\Box^2 [ticket @ M_M \otimes ticket @ W_w \uparrow \otimes 20 \Box @ W_w]]$.

From the above reasoning, the writer can sell a ticket for 20 dollars to meet the request. An instance with $X = M$ is proposed.

W to M: PROPOSE $\Box[20$@M$_M \rightarrow ticket @ M_M \otimes ticket @ W_w \uparrow \otimes 20 \Box @ W_w]$

Given a possible proof of its request formula as described above, the musician will accept the sale

M to W: ACCEPT

The condition of the proposal $\Box^2 20$@M$_M$ suggests that to carry out the sale, the musician needs to pay 20 dollars at the second next time point. If the musician does not have this amount at hand, it can make a commitment to pay such amount. A commitment of $[\Box^2 20$@M$_M] \uparrow$ is formed. As a result of the presence of this commitment, the corresponding formula $[\Box^2 20$@M$_M]$ is available. If the musician still has 15 $\Box$@M$_M$, it only has to make $\Box^2 5$@M$_M$ available and hence a commitment $[\Box^2 5$@M$_M] \uparrow$, because $[\Box^2 20$@M$_M] \uparrow \otimes 15 \Box$@M$_M \vdash [\Box^2 5$@M$_M] \uparrow$.

The proposal can be applied to yield
\[
\Box^2 (ticket @ M_M \otimes ticket @ W_w \uparrow \otimes 20 \Box @ W_w)
\]
As a result, the proposal gives the musician a ticket from the writer, the writer 20 dollars and yields a commitment of the musician to pay 20 dollars.

As the musician has a commitment to pay 20 dollars $[\Box^2 20$@M$_M] \uparrow$, if the musician does not have enough money, it will try to utilize other rules like Rule 1 (selling book) and Rule 2 (selling mp3 music) which derive $5$@M and $25$@M respectively at the current time point, at the next time point or at the second next time point.

3. Sale of a ticket from the artist to the musician

This is similar to the sale of a ticket from the writer.
The sale can start by a request from the musician. It also might start by a proposal from the artist so as to satisfy its commitments to derive some money. Either way, the interactions share a common part, starting from the proposal of the artist.

The musician makes a request for a ticket to the artist

M to A: REQUEST $\Box^2\text{ticket}@M_M$\r

A proof can be constructed using Rule 7 to derive $\Box^2\text{ticket}@M_M$. Other requirements are also analyzed in a similar manner. Hence, the artist can sell his ticket for 25 dollars. An instance of Rule 7 with $X = M$ is proposed.

A to M: PROPOSE $\Diamond [25\$@M_M \rightarrow \text{ticket}@M_M \otimes \text{ticket}@A_A \otimes 25 \Diamond \$@A_A]$

With similar reasoning as in the section of ticket sale from the writer, M to A: ACCEPT

This also requires the musician to pay 25 dollars at the second next time point ($\Box^225\$@M_M$). The musician can make the money amount available by committing to having it ($\Box^225\$@M_M^{-1}$).

Hence, the proposal can be applied. As a result, at the time of application, the proposal gives the musician a ticket from the artist, a commitment to pay 25 dollars and the artist 25 dollars.

4. Sale of a book from the musician to the writer

The sale can start by a request from the writer. It can also start directly from a proposal of Rule 2 from the musician so that the musician can gain some money it possibly needs to buy tickets.

The writer may request from the musician a book.

W to M: REQUEST $\Box^2\text{book}@W_W$\r

To fulfill $\Box^2\text{book}@W_W$, a proof can be constructed from Rule 2 of the musician to derive $\Box^2\text{book}@W_W$.

Consider an application of Rule 2 as follows. Denoting Rule 2 (with the assignment $X = M$) as $\Box [L2 \rightarrow R2]$, we then have

$$
\vdash \Box^2L2 \\
\vdash [L2 \rightarrow R2] \vdash \Box^2R2 \text{ app}
$$

where $\Box^2R2 = \Box^2[\Diamond \text{book}@W_W \otimes \text{book}@M_M \otimes 25 \Diamond \$@M_M]$, which can be split up to make the sub-part $\Box \Diamond \text{book}@W_W$ available to fulfill $\Box^2\text{book}@W_W$ as below.
\( \Box \square \text{book}@W_w \vdash \Box \square \text{book}@W_w \).

The condition for the application of Rule 2 is deriving \( \Box \square L2 = \Box \square 25\$@W_w \).

Given the suitability of Rule 2, the musician proposes the sale of the book for 25 dollars.

\[ \text{M to W: PROPOSE } \Box \square [25\$@W_w \rightarrow \Box \square \text{book}@W_w \otimes \Box \square M_{M_{M}}] \Box \square 25\$@M_{M} \]

\[ \text{W to M: ACCEPT} \]

If the writer does not have 25 dollars at the second next time point, to fulfill the requirement, the writer will make a commitment of \( [\Box \square 25\$@W_w] \). From the proof, after the rule application, a commitment \( \Box \square \text{book}@M_{M_{M}} \) of the musician is also required. The musician has the book which can be sold at the second next time point: \( \Box \square \text{book}@M_{M} \vdash \Box \square \text{book}@M_{M} \), hence fulfilling the commitment \( \Box \square \text{book}@M_{M_{M}} \).

Hence, the proposal can be applied and will give the writer the book \( (\Box \square \text{book}@W_w) \) and a commitment to pay 25 dollars \( (\Box \square 25\$@W_w) \) and the musician 25 dollars \( (\Box \square 25\$@M_{M}) \).

5. Sale of an mp3 player from the writer to the artist

The sale can start by a request from the artist. It can also start directly from a proposal of Rule 6 from the writer so that the writer can gain some money if it has some commitments to derive money.

The artist may ask the writer for the mp3 player.

\[ \text{A to W: REQUEST } (\Box \square 3\text{mp3}\_\text{player}@A_{A}) \]

A proof can be formed out of Rule 6. The proof further requires \( \Box \square 3\text{mp3}\_\text{player}@W_w \).

Rule 6 suggests that the writer offers an mp3 player for 15 dollars.

If the writer already has a commitment to get some money at the second next time point, applying Rule 6 at the second next time point will help it resolve the commitment. On the other hand, if the writer does not yet have such commitments, applying Rule 6 at the third next time point will eventually cause the interaction to fail (as not all of its payment commitments will be resolved).

We consider the case of rule application the second next time point where it is possible to succeed. Rule 6 is assigned with \( X = A \) and proposed to the artist:

\[ \text{W to A: PROPOSE } \Box \square [15\$@A_{A} \rightarrow \Box \square \text{mp3}\_\text{player}@A_{A} \otimes \Box \square \text{mp3}\_\text{player}@W_w] \otimes 15 \Box \square W_w \]

Given that \( \Box \square \text{mp3}\_\text{player}@A_{A} \vdash \Box \square 3\text{mp3}\_\text{player}@A_{A} \), which can resolve the commitment of \( [\Box \square 3\text{mp3}\_\text{player}@A_{A}] \), the artist replies:

\[ \text{A to W: ACCEPT} \]

If the artist does not have 15 dollars at the second next time point, it will make a
commitment:
\[\bigcirc^215\$@A_A\]⊥.

The proposal can hence be carried out to yield an mp3 player for the artist (\(\bigcirc^3\text{mp3}_{\text{player}@A}\)) from the writer, a commitment to pay a debt of 15 dollars (\([\bigcirc^215\$@A_A\]⊥) and 15 dollars for the writer (\(\bigcirc^2\square15\$@W\)).

6. Sale of mp3 music from the musician to the writer

The sale can start by a request from the writer. It can also start from a proposal of Rule 1 from the musician as the musician tries to derive money to buy tickets.

The writer asks the musician for mp3 music.

W to M: REQUEST \([\bigcirc^2\text{mp3}@W_w]\]⊥

A proof of \(\bigcirc^2\text{mp3}@W_w\) uses Rule 1 and \(\bigcirc^25\$@W_w\).

The musician proposes to the writer a sale of each hour of mp3 music for 5 dollars to reply to the request or to gain some money. Rule 1 is assigned with X = M and proposed twice (note that two copies of Rule 1 in the proposal are used as a shorthand for two proposals):

M to W: PROPOSE 2 \(\bigcirc^2[5\$@W_w \rightarrow \square\text{mp3}@W_w \otimes \text{mp3}@M_m^\perp \otimes 5\square@M_m]\]

W to M: ACCEPT

The requirement of the proof of \(\bigcirc^25\$@W_w\) can be resolved if the writer has 5 dollars at the second next time point or makes a commitment: \([\bigcirc^25\$@W_w]\]⊥.

Each instance of the proposal rule can be applied to yield one hour of mp3 music (\(\bigcirc^2\text{mp3}@W_w\)) to the writer from the musician and a commitment to pay 5 dollars (\([\bigcirc^25\$@W_w]\]⊥) of the writer, and 5 dollars for the musician (\(\bigcirc^25\$@M_m\)). We thus have \(\bigcirc^2\text{mp3}@W_w \otimes \bigcirc^2\text{mp3}@W_w \rightarrow \bigcirc^2(\text{mp3}@W_w \otimes \text{mp3}@W_w)\), fulfilling the request for \([\bigcirc^2\text{mp3}@W_w]\]⊥.

7. Sale of mp3 music from the musician to the artist

The sale can start by a request from the artist. It can also start from a proposal of Rule 1 from the musician as the musician tries to derive money to buy tickets.

The artist may ask the musician for a sale of mp3 music.

A to M: REQUEST 2\(\bigcirc^3\text{mp3}@A_A\]⊥

A proof of \(\bigcirc^3\text{mp3}@A_A\) uses Rule 1. The outcome of Rule 1’s application at the third next time point will make the musician failed to resolve all of its payment commitments, which are at the second next time point. As a result, Rule 1 should be applied at the second next time point.
M to A: PROPOSE $2 \odot^2 [5s @ A \rightarrow \Box mp3@A \otimes mp3@[M^M] \otimes 5 \Box s @ M_w]$

As the outcome provides $2 \odot^2 \Box mp3@A \vdash 2 \odot^3 mp3@A$, which fulfills the commitment of $2[\odot^3 mp3@A]^\perp$

A to M: ACCEPT

The interaction continues in a similar manner as in the sale of mp3 from the musician to the writer.

8. The writer listening to music

Given that the writer has obtained mp3 music from the musician, it tries to apply Rule 3 to derive music. There are two cases depending on the availability of resources.

**Case 1**: if the writer has sold its mp3 player to the artist, it can not take the option of using the mp3 player in Rule 3. Therefore, the writer will apply Rule 4, given two hours of mp3 music available ($\odot^2 mp3@[W_w]$):

$2 \odot^2 mp3@[W_w] \otimes \Box [2mp3@[W_w] \rightarrow CD@[W_w]] \vdash$
$2 \odot^2 mp3@[W_w] \otimes \odot^2 [2mp3@[W_w] \rightarrow CD@[W_w]] \vdash$
$\odot^2 CD@[W_w]$

Rule 3 can be applied to derive $\odot^2 music@[W_w]$. Hence the writer can listen to the music as desired.

**Case 2**: if the writer has not sold the mp3 player, it can take the option of listening on CD player as above or choose to listen on mp3 player. Rule 3 can be applied to derive $\odot^2 music@[W_w]$ as above. However, if the writer chooses to do so, it cannot later sell the mp3 player to the artist to gain enough money to buy the book from the musician and hence satisfy its commitment of reading a book. Also, the artist will not be able to fulfill its commitment of carrying out a performance, which requires an mp3 player purchased from the writer. Taking this option therefore causes the interaction to fail.

The interaction ends after Step 7. After Step 8, all of the agents’ goals and commitments are achieved. The results of the interaction are further considered in the following section on sequences of interactive actions.

8.1.3 Sequences of Interactive Actions

We have demonstrated the construction of an interaction among the agents following the specification, based on the assumed interaction model. The interaction can be viewed as a sequence of all the steps discussed above. While Step 1 refers to the initial state of agents and
Step 8 refers to internal agent reasoning about fulfillment of the commitments, the interaction from Step 2 to Step 7 can be carried out in any order by the agents to achieve their goals. In fact, the interaction can start with any of the agents making a request for its goals. Hence, any of the sales can be the first one.

In this section, we examine the different sequences that are allowed by the specification. We abbreviate an interaction as a list of messages exchanged as follows. The status of each agent is also shown with respect to their resulting resources and commitments after each step. We will examine an interaction as a sequence of Steps 1, 2, 3, 4, 5, 6, 7, and 8. We then consider another arbitrary order of sales, such as the order of Steps 1, 5, 7, 3, 2, 6, 4 and Step 8. This later sequence also leads to the same end results.

**Interaction as a sequence of Steps 1, 2, 3, 4, 5, 6, 7, and 8**

1. **Initial States**

   **Musician:**
   Resources: $\eta \square mp3@M_M \otimes \square book@M_M \otimes 15 \square \$@M_M$.
   Commitments: $[\bigcirc^2 ticket@M_M]^\perp \otimes [\bigcirc^2 ticket@M_M]^\perp$.
   Rules used: none.

   **Writer:**
   Resources: $\& \sum_{i \geq 0}^3 (i^\prime ticket@W_w) \otimes \square CD\_player@W_w \otimes mp3\_player@W_w$.
   Commitments: $[\bigcirc^2 mp3@W_w]^\perp \otimes [CD\_player@W_w \oplus mp3\_player@W_w]^\perp \otimes [\bigcirc^2 book@W_w]^\perp$.
   Note that the choice between $[\bigcirc^2 mp3\_player@W_w]^\perp$ and $[\bigcirc^2 CD\_player@W_w]^\perp$ can be expressed as $[\bigcirc^2 mp3\_player@W_w] \otimes [\bigcirc^2 CD\_player@W_w]^\perp$.
   Rules used: none.

   **Artist:**
   Resources: $\& \sum_{i \geq 0}^3 i^\prime ticket@A_A$.
   Commitments: $2(\bigcirc^3 mp3@A_A]^\perp \otimes [\bigcirc^3 mp3\_player@A_A]^\perp$.
   Rules used: 8.

2. **Sale of a ticket from the writer to the musician**

   M to W: REQUEST $[\bigcirc^2 ticket@M_M]^\perp$
   W to M: PROPOSE $\square [20\$@M_M \rightarrow ticket@M_M \otimes ticket@W_w \otimes 20 \square \$@W_w]$.
   M to W: ACCEPT

   **Musician:**
   Resources: $\eta \square mp3@M_M \otimes \square book@M_M$.
   Commitments: $[\bigcirc^2 ticket@M_M]^\perp \otimes [\bigcirc^2 5\$@M_M]^\perp$. 
Rules used: none.

**Writer:**

Resources: $\square CD_{\text{player}}@W_w \otimes mp3_{\text{player}}@W_w \otimes 20 \square \$@W_w$.

Commitments: $[O^22mp3@W_w]^{-1} \otimes [CD_{\text{player}}@W_w \oplus mp3_{\text{player}}@W_w]^{-1} \otimes [O^2\text{book}@W_w]^{-1}$.

Rules used: 5.

**Artist:**

Resources: $\exists_i^n (O^i \text{ticket}@A_A)$.

Commitments: $2[O^3mp3@A_A]^{-1} \otimes [O^3mp3_{\text{player}}@A_A]^{-1}$.

Rules used: 8.

3. **Sale of a ticket from the artist to the musician**

M to A: REQUEST $[O^2\text{ticket}@M_M]^{-1}$

A to M: PROPOSE $\square [25\$@M_M \rightarrow \text{ticket}@M_M \otimes \text{ticket}@A_A^{-1} \otimes 25 \square \$@A_A]$

M to A: ACCEPT

**Musician:**

Resources: $\eta \square mp3@M_M \otimes \text{book}@M_M$.

Commitments: $[O^23\$@M_M]^{-1}$.

Rules used: none.

**Writer:**

Resources: $\square CD_{\text{player}}@W_w \otimes mp3_{\text{player}}@W_w \otimes 20 \square \$@W_w$.

Commitments: $[O^22mp3@W_w]^{-1} \otimes [CD_{\text{player}}@W_w \oplus mp3_{\text{player}}@W_w]^{-1} \otimes [O^2\text{book}@W_w]^{-1}$.

Rules used: 5.

**Artist:**

Resources: $25 \square \$@A_A$.

Commitments: $2[O^3mp3@A_A]^{-1} \otimes [O^3mp3_{\text{player}}@A_A]^{-1}$.

Rules used: 7, 8.

4. **Sale of a book from the musician to the writer**

(OPTIONAL)W to M: REQUEST $[O^2\text{book}@W_w]^{-1}$

M to W: PROPOSE $\square [25\$@W_w \rightarrow \square \text{book}@W_w \otimes \text{book}@M_M^{-1} \otimes 25 \square \$@M_M]$

W to M: ACCEPT

**Musician:**

Resources: $\eta \square mp3@M_M$.

Commitments: $[O^25\$@M_M]^{-1}$.

Rules used: 2.

**Writer:**
Resources: \(\Box CD_{\text{player}}@W_w \otimes \Box mp3_{\text{player}}@W_w\).
Commitments: \(\bigcirc^{2}2mp3@W_w\)' \(\uparrow \otimes [CD_{\text{player}}@W_w \oplus mp3_{\text{player}}@W_w]\)' \(\uparrow \otimes [\bigcirc^{2}5@$@W_w]\)'.
Rules used: 5.

Artist:
Resources: \(25 \bigcirc @$@A_A\).
Commitments: \(2[\bigcirc^{3}mp3@A_A]\)' \(\uparrow \otimes [\bigcirc^{3}mp3_{\text{player}}@A_A]\)'.
Rules used: 7, 8.

5. Sale of an mp3 player from the writer to the artist
(OPTIONAL) A to W: REQUEST \([\bigcirc^{3}mp3_{\text{player}}@A_A]\)'\(\downarrow\)
W to A: PROPOSE \(\bigcirc^{2}[15@$@A_A \rightarrow \Box mp3_{\text{player}}@A_A \otimes mp3_{\text{player}}@W_w\)' \(\uparrow \otimes 15 \Box @$@W_w\)
A to W: ACCEPT

Musician:
Resources: \(\eta \bigcirc mp3@M_M\).
Commitments: \(\bigcirc^{2}5@$@M_M\)'\(\downarrow\).
Rules used: 2.

Writer:
Resources: \(\Box CD_{\text{player}}@W_w \otimes 10 \Box @$@W_w\).
Commitments: \(\bigcirc^{2}2mp3@W_w\)' \(\uparrow \otimes [CD_{\text{player}}@W_w \oplus mp3_{\text{player}}@W_w]\)'\(\downarrow\).
Rules used: 5, 6.

Artist:
Resources: \(10 \bigcirc @$@A_A\).
Commitments: \(2[\bigcirc^{3}mp3@A_A]\)'\(\downarrow\).
Rules used: 7, 8.

6. Sale of mp3 music from the musician to the writer
(OPTIONAL) W to M: REQUEST \([\bigcirc^{2}2mp3@W_w]\)'\(\downarrow\)
M to W: PROPOSE \(2 \bigcirc^{2} [5@$@W_w \rightarrow \Box mp3@W_w \otimes mp3@M_M\)' \(\uparrow \otimes 5 \Box @$@M_M\)
W to M: ACCEPT

Musician:
Resources: \((\eta - 2) \bigcirc mp3@M_M \otimes 5 \bigcirc @$@M_M\).
Note that, for a shorthand, we denote \(\eta - 2\) as the number (not the subtraction operation) that indicates the remaining copies of the formula \(\Box mp3@M_M\). We make use of this notation for the rest of the example.
Commitments: none.
Rules used: 1, 2.
Writer:
Resources: $\Box CD_{\text{player}@W_w}$.  
Commitments: $[CD_{\text{player}@W_w} \oplus mp3_{\text{player}@W_w}]^\perp$.  
Rules used: 5, 6.

Artist:
Resources: $10 \Box \$@A_A$.  
Commitments: $2[\bigcirc^3 mp3@A_A]^\perp$.  
Rules used: 7, 8.

7. Sale of mp3 music from the musician to the artist  
A to M: REQUEST $2[\bigcirc^3 mp3@A_A]^\perp$  
M to A: PROPOSE $2 \bigcirc^2 [5\$@A_A \rightarrow \Box mp3@A_A \otimes mp3@M^\perp \otimes 5 \Box \$@M_M]$  
A to M: ACCEPT  
Resources: $(\eta - 4) \Box mp3@M_M \otimes 15 \Box \$@M_M$.  
Commitments: none.  
Rules used: 1, 2.

Musician:
Resources: $(\eta - 4) \Box mp3@M_M \otimes 15 \Box \$@M_M$.  
Commitments: none.  
Rules used: 1, 2.

Writer:
Resources: $\Box CD_{\text{player}@W_w}$.  
Commitments: $[CD_{\text{player}@W_w} \oplus mp3_{\text{player}@W_w}]^\perp$.  
Rules used: 5, 6.

Artist:
Resources: none.  
Commitments: none.  
Rules used: 7, 8.

8. Listening to music at the writer  
Musician:
Resources: $(\eta - 4) \Box mp3@M_M \otimes 15 \Box \$@M_M$.  
Commitments: none.  
Rules used: 1, 2.

Writer:
Resources: none.  
Commitments: none.  
Rules used: 3, 4, 5, 6.

Artist:
Resources: none.
Commitments: none.
Rules used: 7, 8.

The interaction in the order of sessions: 1, 2, 3, 4, 5, 6, 7, and 8 ends and meets the criteria for a successful run, as there are no pending commitments. Hence, the interaction is successful.

Interaction as a sequence of Steps 1, 5, 7, 3, 2, 6, 4 and 8

1. Initial States

Musician:
Resources: $\eta \Box mp3@M_M \otimes \Box book@M_M \otimes 15 \Box @M_M$.
Commitments: $[O^2ticket@M_M]^\perp \otimes [O^2ticket@M_M]^\perp$.
Rules used: none.

Writer:
Resources: $\& \{i \geq 1 (O\Box ticket@W_w) \otimes \Box CD_player@W_w \otimes \Box mp3_player@W_w.$
Commitments: $[O^2mp3@W_w]^\perp \otimes [CD_player@W_w \oplus mp3_player@W_w]^\perp \otimes [O^2book@W_w]^\perp$.

Rules used: 6, 8.

Artist:
Resources: $\& \{i \geq 1 (O\Box ticket@A_A) \}$.
Commitments: $[O^3mp3@A_A]^\perp \otimes [O^3mp3_player@A_A]^\perp$.

Rules used: none.

5. Sale of an mp3 player from the writer to the artist

A to W: REQUEST $[O^3mp3_player@A_A]^\perp$

W to A: PROPOSE $O^2[15@A_A \rightarrow \Box mp3_player@A_A \otimes mp3_player@W_w \oplus 15 \Box @W_w]$

A to W: ACCEPT

Musician:
Resources: $\eta \Box mp3@M_M \otimes \Box book@M_M \otimes 15 \Box @M_M$.
Commitments: $[O^2ticket@M_M]^\perp \otimes [O^2ticket@M_M]^\perp$.
Rules used: none.

Writer:
Resources: $\& \{i \geq 1 (O\Box ticket@W_w) \otimes \Box CD_player@W_w \otimes 15 \Box @W_w$.
Commitments: $[O^2mp3@W_w]^\perp \otimes [CD_player@W_w \oplus mp3_player@W_w]^\perp \otimes [O^2book@W_w]^\perp$.

Rules used: 6.

Artist:
Resources: $\& \{i \geq 1 (O\Box ticket@A_A) \}.$
Commitments: $2[O^3\text{mp3@A}_A]^\bot \otimes [15\@A_A]^\bot$.

Rules used: 8.

7. Sale of mp3 music from the musician to the artist

A to M: REQUEST $2[O^3\text{mp3@A}_A]^\bot$

M to A: PROPOSE $2 O^2 [5\@A_A \rightarrow \Box \text{mp3@A}_A \otimes \text{mp3@M}_M \otimes 5 \Box \@M_M]$.

A to M: ACCEPT

Musician:

Resources: $(\eta - 2) \Box \text{mp3@M}_M \otimes \Box \text{book@M}_M \otimes 25 \Box \@M_M$.

Commitments: $[O^2\text{ticket@M}_M]^\bot \otimes [O^2\text{ticket@M}_M]^\bot$.

Rules used: 1.

Writer:

Resources: $\& 3_{i \geq 0} (\bigotimes i \text{ticket@W}_W) \otimes \Box \text{CD_player@W}_W \otimes 15 \Box \@W_W$.

Commitments: $[O^2\text{mp3@W}_W]^\bot \otimes [\Box \text{CD_player@W}_W \oplus \text{mp3_player@W}_W]^\bot \otimes 25 \Box \@W_W$.

Rules used: 6.

Artist:

Resources: $\& 3_{i \geq 0} (\bigotimes i \text{ticket@A}_A$.

Commitments: $[25\@A_A]^\bot$.

Rules used: 8.

3. Sale of a ticket from the artist to the musician (OPTIONAL)

M to A: REQUEST $[O^2\text{ticket@M}_M]^\bot$.

A to M: PROPOSE $\Box [25\@M_M \rightarrow \text{ticket@M}_M \otimes \text{ticket@A}_A^\bot \otimes 25 \Box \@A_A]$.

M to A: ACCEPT

Musician:

Resources: $(\eta - 2) \Box \text{mp3@M}_M \otimes \Box \text{book@M}_M$.

Commitments: $[O^2\text{ticket@M}_M]^\bot$.

Rules used: 1.

Writer:

Resources: $\& 3_{i \geq 0} (\bigotimes i \text{ticket@W}_W) \otimes \Box \text{CD_player@W}_W \otimes 15 \Box \@W_W$.

Commitments: $[O^2\text{mp3@W}_W]^\bot \otimes [\Box \text{CD_player@W}_W \oplus \text{mp3_player@W}_W]^\bot \otimes 25 \Box \@W_W$.

Rules used: 6.

Artist:

Resources: none.

Commitments: none.

Rules used: 7, 8.
2. Sale of a ticket from the writer to the musician

M to W: REQUEST $[\mathit{2\ticket}@M_M]^\perp$

W to M: PROPOSE $\square[20\$@M_M \to \ticket@M_M \otimes \ticket@W_W \otimes 20 \square \$@W_W]$

M to W: ACCEPT

Musician:
Resources: $(\eta - 2) \square \text{mp3}@M_M \otimes \square \text{book}@M_M$.
Commitments: $[\mathit{2\20\$}@M_M]^\perp$.
Rules used: 1.

Writer:
Resources: $\square \text{CD}\_\text{player}@W_W \otimes 35 \square \$@W_W$.
Commitments: $[\mathit{2\2\text{mp3}@W_W}]^\perp \otimes [\text{CD}\_\text{player}@W_W \oplus \text{mp3}\_\text{player}@W_W]^\perp \otimes [\mathit{2\text{book}@W_W}]^\perp$.
Rules used: 5, 6.

Artist:
Resources: none.
Commitments: none.
Rules used: 7, 8.

6. Sale of mp3 music from the musician to the writer
(OPTIONAL) W to M: REQUEST $[\mathit{2\text{mp3}@W_W}]^\perp$

M to W: PROPOSE $2 \mathit{2}[5\$@W_W \to \square \text{mp3}@W_W \otimes \text{mp3}@M_M \oplus 5 \square \$@M_M]$.

W to M: ACCEPT

Musician:
Resources: $(\eta - 4) \square \text{mp3}@M_M \otimes \square \text{book}@M_M$.
Commitments: $[\mathit{2\10\$}@M_M]^\perp$.
Rules used: 1.

Writer:
Resources: $\square \text{CD}\_\text{player}@W_W \otimes 25 \square \$@W_W$.
Commitments: $[\text{CD}\_\text{player}@W_W \oplus \text{mp3}\_\text{player}@W_W]^\perp \otimes [\mathit{2\text{book}@W_W}]^\perp$.
Rules used: 5, 6.

Artist:
Resources: none.
Commitments: none.
Rules used: 7, 8.

4. Sale of a book from the musician to the writer
(OPTIONAL) W to M: REQUEST $[\mathit{2\text{book}@W_W}]^\perp$
Concert Ticket Example

M to W: PROPOSE $25\oplus W \to [\text{book}@W \otimes \text{book}@M] \lands 25 \oplus W \oplus M$

W to M: ACCEPT

Musician:
Resources: $(\eta - 4) \oplus mp3@M \otimes 15 \oplus M \oplus M$
Commitments: none.
Rules used: 1, 2.

Writer:
Resources: $\square C D \_\text{player}@W$
Commitments: $[C D \_\text{player}@W \oplus mp3\_\text{player}@W] \perp$.
Rules used: 5, 6.

Artist:
Resources: none.
Commitments: none.
Rules used: 7, 8.

8. Listening to music at the writer

Musician:
Resources: $(\eta - 4) \oplus mp3@M \otimes 15 \oplus M \oplus M$
Commitments: none.
Rules used: 1, 2.

Writer:
Resources: none.
Commitments: none.
Rules used: 3, 4, 5, 6.

Artist:
Resources: none.
Commitments: none.
Rules used: 7, 8.

The interaction with a sequence of Steps 1, 5, 7, 3, 2, 6, 4 and 8 ends successfully as all the required commitments are fulfilled.

As can be seen, all the interactions with different sequences of sections end up successfully if the writer does not make the choice of listening to music on the mp3 player, which corresponds to using the resources $\sqcup [2mp3@W \oplus mp3\_\text{player}@W] \perp$ to apply Rule 3.

In the case where the writer chooses to listen to music on its mp3 player, the corresponding interaction will fail. When the writer uses its mp3 player, the mp3 player is no longer available.
Concert Ticket Example

for sale to the artist. As a result, the writer does not have enough money to buy book from
the musician and the artist can not fulfill its commitment of a performance. The interaction
then fails.

8.1.4 Integration of Changes

We now consider some changes in the environment for showing an integration of changes in
protocol execution. Exemplar changes are the addition of an mp3 player available for use at
the artist ($\text{mp3}\_\text{player} @ A_A$) and 20 dollars for the writer ($20 \square $@W_w$) during a protocol
execution. We can pick a protocol run with a random sequence of Steps 2 to 7 (with Step
1 beginning the sequence and Step 8 ending it). However, for showing the most effects and
for familiarity with the example, we consider such changes in Step 4 of the protocol as a
sequence of Steps 1, 2, 3, 4, 5, 6, 7, and 8.

At the end of Step 4, the states of the agents are:

4. Sale of a book from the musician to the writer

(OPTIONAL) $W$ to $M$: REQUEST $[\bigcirc^2 \text{book} @ W_w]^\perp$

$M$ to $W$: PROPOSE $\square [25 \square \$ W_w \rightarrow \square \text{book} @ W_w \odot \text{book} @ M_M \otimes 25 \square \$ M_M]$

$W$ to $M$: ACCEPT

Musician:

Resources: $\eta \square \text{mp3} @ M_M$.

Commitments: $[\bigcirc^2 5 \square \$ M_M]^\perp$.

Rules used: 2.

Writer:

Resources: $\square \text{CD}\_\text{player} @ W_w \otimes \square \text{mp3}\_\text{player} @ W_w \otimes 20 \square \$ W_w$ (newly included).

Commitments: $[\bigcirc^2 2 \text{mp3} @ W_w]^\perp \otimes [\text{CD}\_\text{player} @ W_w \oplus \text{mp3}\_\text{player} @ W_w]^\perp \otimes [\bigcirc^2 5 \square \$ W_w]^\perp$.

Rules used: 5.

Artist:

Resources: $25 \square \$ A_A \otimes \square \text{mp3}\_\text{player} @ A_A$ (newly included).

Commitments: $2[\bigcirc^3 \text{mp3} @ A_A]^\perp \otimes [\bigcirc^3 \text{mp3}\_\text{player} @ A_A]^\perp$.

Rules used: 7, 8.

The next step in the sequence is Step 5 which is a sale of an mp3 player from the
writer to the artist. However, because $\square \text{mp3}\_\text{player} @ A_A \vdash \bigcirc^3 \text{mp3}\_\text{player} @ A_A$, the resource
$\square \text{mp3}\_\text{player} @ A_A$ can satisfy the commitment $[\bigcirc^3 \text{mp3}\_\text{player} @ A_A]^\perp$. Therefore, the artist
does not need to request an mp3\_player from any other agent. Moreover, the writer has
20\Box \Diamond \$@W_w$, and therefore the commitment $[\Diamond 2\Box 5\Box \Diamond \$@W_w] \perp$ is fulfilled and $15\Diamond \$@W_w$ remains. Thus, the writer does not have a monetary commitment as an incentive to propose Rule 6. Step 5 in the sequence is then skipped.

6. Sale of mp3 music from the musician to the writer (OPTIONAL) W to M: REQUEST $[\Diamond 2\Box 5\Box \Diamond \$@W_w] \perp$

M to W: PROPOSE $2 \Box [5\Box \Diamond \$@W_w \rightarrow \Box mp3@W_w \otimes mp3@M_M \otimes 5 \Diamond \$@M_M]$

W to M: ACCEPT

Musician:
Resources: $(\eta - 2) \Box mp3@M_M \otimes 5 \Diamond \$@M_M$.
Commitments: none.
Rules used: 1, 2.

Writer:
Resources: $\Box CD_player@W_w \otimes \Box mp3_player@W_w \otimes 5 \Diamond \$@W_w$.
Commitments: $[CD_player@W_w \oplus mp3_player@W_w] \perp$.
Rules used: 5.

Artist:
Resources: $25 \Diamond \$@A_A$.
Commitments: $2[\Box 5\Box mp3@A_A] \perp$.
Rules used: 7, 8.

7. Sale of mp3 music from the musician to the artist
A to M: REQUEST $2[\Box 5\Box mp3@A_A] \perp$

M to A: PROPOSE $2 \Box [5\Box \Diamond \$@A_A \rightarrow \Box mp3@A_A \otimes mp3@M_M \otimes 5 \Diamond \$@M_M]$

A to M: ACCEPT

Musician:
Resources: $(\eta - 4) \Box mp3@M_M \otimes 15 \Diamond \$@M_M$.
Commitments: none.
Rules used: 1, 2.

Writer:
Resources: $\Box CD_player@W_w \otimes \Box mp3_player@W_w \otimes 5 \Diamond \$@W_w$.
Commitments: $[CD_player@W_w \oplus mp3_player@W_w] \perp$.
Rules used: 5.

Artist:
Resources: $15 \Diamond \$@A_A$.
Commitments: none.
Rules used: 7, 8.

8. Listening to music at the writer

The writer can use either the CD player or the mp3 player since it has both. For simplicity, we show the case that the writer uses CD player.

Musician:
Resources: \((\eta - 4) \Box mp3@M_M \otimes 15 \Box $@M_M\).
Commitments: none.
Rules used: 1, 2.

Writer:
Resources: \(\Box mp3\_player@W_W \otimes 5 \Box $@W_W\).
Commitments: none.
Rules used: 5.

Artist:
Resources: 15 \(\Box $@A_A\).
Commitments: none.
Rules used: 7, 8.

As there are no pending commitments, the interaction ends successfully as a sequence of Steps 1, 2, 3, 4, 6, 7, and 8, without Step 5. The protocol run demonstrates that agents can take advantage of changes from outside (like the addition of resources) during the execution of the protocol to skip some parts of the protocol.

8.1.5 Summary of Interactions

In the case where the writer chooses to listen to music on its CD player, the various interactions have the following properties.

Throughout these interaction sessions, the musician has the original commitments: \([\Box 2\_ticket@M_M]^\perp\) and \([\Box 2\_ticket@M_M]^\perp\) removed. The purchases of tickets from the writer and from the artist require from the musician the commitments of \([\Box 2\_20\$@M_M]^\perp\) and of \([\Box 2\_25\$@M_M]^\perp\) respectively. The commitments then together become \([\Box 2\_45\$@M_M]^\perp\). On the other hand, the sales of mp3 music to the artist and the writer give the musician \(\Box 2\_10\$@M_M\) and \(\Box 2\_10\Box $@M_M\), and the sale of the book gives the musician \(\Box 2\_25\Box $@M_M\), which together yield \(\Box 2\_45 \Box $@M_M\). This money amount can successfully remove the musician’s commitment of \([\Box 2\_45\$@M_M]^\perp\) as \(\Box 2\_45 \Box $@M_M \otimes [\Box 2\_45\$@M_M]^\perp \vdash \bot\).

The writer, toward the end of the interaction, will have the original commitment of
Temporal Linear Logic (TLL) Modeling

\[ [O^2_{music}@W_w]^\perp \otimes [O^2_{book}@W_w]^\perp \text{ removed.} \] The writer also has \([O^2_{25\$}@W_w]^\perp\) from the purchase of the book and \([O^2_{10\$}@W_w]^\perp\) from the purchase of two hours mp3 music. Together, they make up a commitment: \([O^2_{35\$}@W_w]^\perp\). The writer also has \([O^2_{20\$}@W_w]\) from the sale of the ticket. Similarly, the writer has \([O^2_{15\$}@W_w]\) from the sale of the mp3 player. The writer will have \([O^2_{35\$}@W_w]\) and hence can remove the debt.

The artist has the commitment of \([O^3_{mp3\_player}@A_A]^\perp\) and \([O^3_{2mp3}@A_A]^\perp\) eventually removed. The artist has a commitment of \([O^2_{15\$}@A_A]^\perp\) from the purchase of the mp3 player and a commitment of \([O^2_{10\$}@A_A]^\perp\) from the purchase of 2 hours of mp3 music. However, the artist gets from the sale of the ticket \([O^2_{25\$}@A_A]\). The commitments can be fulfilled by the money available from the sale.

Hence, given after all the interaction sessions, no commitments involved at any agents are pending, the interaction is successful.

In the case where the writer chooses to listen to music on its mp3 player, the corresponding interactions have similar properties, except that in the end, there is a pending commitment \([O^2_{mp3\_player}@W_w]^\perp\) at the writer, which makes the interaction fail.

### 8.2 Temporal Linear Logic (TLL) Modeling

In this section, we present reflective discussions on our TLL modeling framework. Our modeling of agent interaction is resource-centric and reflects a view of agent interaction in terms of resource utilization and actions.

We discuss the modeling of various interaction concepts based on temporal linear logic as introduced in Chapters 3 and 4 (which now is referred to as TLL modeling) from a resource-conscious view. We also use the concert ticket example (abbreviated as CT example) to illustrate our discussions.

#### 8.2.1 Modeling of Resources

The properties of interest for resources are when they are produced, when they are used and when they persist. Being produced can be readily captured by an introduction of the formula representing the resource and persisting means that the formula representing resource is left untouched until the resource is used. The use of linear logic and hence TLL in our modeling makes it quite natural to capture the use of resources. Because TLL treats formulas as resources, resources can be modeled as simply as propositional formulas. Consumption of
resources can be naturally modeled as the removal of the appropriate formula. TLL allows the removal of formulas via the use of linear implication, such as $A \otimes A \rightarrow \bot \vdash \bot$. In classical logic or modal logic, once formulas are introduced, it is hard to remove them and hence other approaches to modeling resources using these logics require operations to change the values of the formulas to reflect that the resource formula is no longer available.

In CT example, resources like money, books, music, mp3 songs, CD players, and so forth have been captured successfully by their respective TLL formulas. Whenever these resources are used, their formulas are removed as part of the application of rules. For example, an application of Rule 3 removes formulas of either (CD and CD player) or (mp3 song and mp3 player). Similarly, applications of Rules 4 and 8 remove mp3 song and (mp3 song together with mp3 player) respectively. Conversely, applications of Rules 3, 4, and 8 introduce new resources, such as music (for listening) and CD and music (for performance) which have not existed before.

Another important aspect of modeling resources is to allow direct handling of the quantity of resources. Unlike classical logic and modal logic, linear logic and hence TLL makes a distinction between having two copies of the same formula and only having one. This makes it easy in our modeling to express quantities of resources (by a multiplicative conjunction of copies of the same formula). For example, five dollars can be expressed as $\square \$ \otimes \square \$ \otimes \square \$ \otimes \square \$. Moreover, changes in resource locations and ownership are also reflected via applications of Rules 1, 2, 4, 6, and 7. Money are transferred between agents in accordance with the sale of goods. In particular, the correct amounts are credited to the sellers while commitments to pay are attributed to the buyers. New instances of resource formulas such as $\lozenge^2 mp3@A_A$, $\lozenge^2 mp3@W_w$, $\lozenge^2 ticket@M_M$, and $\lozenge^2 mp3_player@A_A$, are derived to reflect that these resources are now available and owned by these agents while corresponding commitments such as $[\lozenge^2 mp3@M_M]^\perp$, $[\lozenge^2 mp3@M_M]^\perp$, $[\lozenge^2 ticket@A_A]^\perp$, and $[\lozenge^2 mp3_player@W_w]^\perp$ are formed to remove those resources from the old owners and locations.

### 8.2.2 Modeling Capabilities and Actions

Modeling capabilities and actions can be done by modeling the effects of them. Two important kinds of effects of capabilities and actions in agent interaction are resource transformations and state updates. While a resource transformation can be regarded as a consumption of some resources followed by a production of others, a state update can be regarded as a
replacement of the old state by a new state. Such effects are naturally captured by the use of linear implication where the removal of old resources or the old state leads to the introduction of new resources or a new state. Other approaches using classical logic or modal logic require some further operations to change the values of old formulas appropriately.

In the CT example, the capabilities of agents are encoded via rules that agents can perform at any time. The capabilities are to transform resources like Rules 3, 5, and 8 or relocate resources to other owners like Rules 1, 2, 4, 6, and 7. These capabilities can be used as services to oneself, like Rules 3, 4, and 8 or to other agents (and hence are also pre-commitments) like Rules 1, 2, 5, 6, and 7.

8.2.3 Modeling Base Commitments

A base commitment is a promise to bring about some resources or to perform some actions. On the one hand, it is important to capture the content of the promise as the desired resources and/or actions. On the other hand, the modeling should also differentiate between what is promised and the promise itself.

These aspects are naturally reflected in the modeling of a base commitment as a negation of the formulas of what must be provided (sharing the same content). Moreover, unlike other logics like classical logic and modal logic, in linear logic and hence TLL, positive formulas and their negations can be thought as having a mutual causal relationship in that the presence of a commitment and the resources needed to fulfill it causes the removal of both, as in \( \Gamma \otimes [\Gamma]^\perp = \Gamma \otimes \Gamma \rightarrow \bot \vdash \bot \). This means that the presence of what must be provided causes the commitment to be fulfilled and hence removed. Similarly, the presence of a base commitment causes the desired resources and/or actions to be consumed and/or carried out. Also, the relationship between a base commitment and what is required to fulfill it is also captured naturally in the logic modeling as one to one, which is important as it ensures that anything more than necessary – more commitments or more resources and actions – are left untouched.

It can be seen that the duality between a base commitment and what must be provided is matched by the duality between positive formulas and negative formulas in TLL. This ensures that the duality between is maintained and respected in agent reasoning. Indeed, a base commitment and what must be provided can be expressed as two sides of a derivation in a sequent like

\[
what \text{ must be provided} \vdash base \text{ commitment}
\]
The inference rules of TLL and of choice calculus are in place to manipulate these sequents while preserving the duality between them.

Furthermore, the task of fulfilling base commitments can then be reduced to simply removing negative formulas, which in turn, due to being dual to positive formulas, is essentially deriving the corresponding positive formulas. In the CT example, there are some commitments like $[O^2\text{ticket}@M_m]^\perp$, $[O^2\text{20$b}@M_m]^\perp$ at the musician or $[O^2\text{book}@W_w]^\perp$, $[O^2\text{25$b}@W_w]^\perp$ at the writer, or $[O^2\text{15$b}@A_a]^\perp$ at the artist which can not be fulfilled by the owner agents alone. Other commitments like $[O^2\text{music}@W_w]^\perp$, and $[O^2\text{perform}@A_a]^\perp$ can be derived internally but require extra commitments which in turn can not be derived internally. Trying to remove these negative formulas exerts a pressure on the owner agents to seek relevant resources and actions by making requests or proposing to other agents any of their interaction rules which can derive these commitments. For example, regarding the commitment of $[O^2\text{20$b}@M_m]^\perp$, the musician can propose Rule 1 or 2 to sell mp3 music or a book to gain some money. Similarly, the commitments of the artist ($[O^2\text{5$b}@A_a]^\perp$) and the writer ($[O^2\text{25$b}@W_w]^\perp$) resulting from purchasing music from the musician will prompt them to propose selling their tickets.

In addition, the duality property can be further explored in the context of proactive reasoning. When an agent considers the consequences of having a resource or doing an action, named $A$, it can create a base commitment $A^\perp$ to derive that resource or to do that action, and continue further planning and reasoning with the presence of $A$. This can be regarded as simply adding $A \otimes A^\perp$. Because the concurrent existence of a formula and its dual does not produce anything significant, like $A \otimes A^\perp \vdash \bot$, assuming their concurrent presence then does not distort the overall outcomes, like $[A \otimes A^\perp], \Gamma \vdash \Gamma \otimes \bot$. Moreover, by making the outcomes of base commitments available via the above-mentioned mechanism and reasoning about them with respect to fulfilling their goals, agents can reason about base commitments, such as which ones to make, and when to make them. This feature is also evident in the CT example. For example, in order to satisfy the conditions of Rule 1, the artist makes $O^2\text{5$b}@A_a$ available by also making a commitment of $[O^2\text{5$b}@A_a]^\perp$. Similarly, the writer makes a commitment of $[O^2\text{25$b}@W_w]^\perp$ to have $O^2\text{25$b}@W_w$ available to satisfy the condition of Rule 2.

Moreover, representing base commitments as negative formulas is a stepping stone toward providing a mechanism for expressing their dynamic relationships and manipulating them. Existing logic connectives like $\otimes$, $\&$, $\oplus$, $\neg$ are readily available for describing the relationships among base commitments. Base commitments can also be embedded in other
commitments. For example, an embedded commitment might be \((A \otimes com^\perp)^\perp\), where \(com^\perp\) is a base commitment.

However, modeling base commitments as negative formulas causes some difficulties in handling them. A positive formula \(A\) is equivalent to its double negation form: \(A \equiv (A^\perp)^\perp\). Hence, a resource or action \(A\) might be interpreted as a base commitment to bring about \(A^\perp\), which in turn is a base commitment to derive \(A\). On the one hand, this interpretation drives the agent to utilize the resource or action \(A\) by matching it with a commitment to derive \(A - A^\perp\). On the other hand, when \(A^\perp\) is not readily available, this interpretation complicates the utilization of the resource or action \(A\). Therefore, extra caution must be taken in the implementation of base commitments as negative formulas to avoid such translation from \(A\) to \((A^\perp)^\perp\).

Another limitation in our modeling of base commitments as negative formulas concerns breakable commitments. In Section 3.3.2, a modeling of breakable base commitments was introduced by including an internal choice on the final outcome inside the base commitment’s formula. This modeling captures breakable commitments in the cases when an agent makes a base commitment and retains the choice whether to fulfill it or not. The modeling however does not capture the cases where the agent does not intend to break the commitment when making it. This remains an item of future work.

Moreover, our modeling does not make a distinction between goals and base commitments. Where a distinction should be made between them, treating base commitments as negative formulas means that goals must be modeled differently. This is also an item of future work.

### 8.2.4 Modeling Conditional Commitments

A conditional commitment is essentially one or more base commitments that require prior conditions. It is important to capture the relationship between the conditions and the commitment part. In particular, fulfillment of the conditions causes the commitment to be made. While it is possible to model similar rules in classical logic and modal logic, in TLL, \(conditions \rightarrow commitments\) naturally expresses that one provision of the conditions always causes one occurrence of the commitment (whereas classically this needs not be the case). Also, our modeling expresses each conditional commitment naturally as a one-time relationship between the conditions and the commitments, which prevents the case where the agent has to commit whenever the conditions are provided.

The modeling of conditional commitments is also shown in the example. Rules 1, 2, 5, 6,
and 7 are also considered as potential conditional commitments of the corresponding owner agents. Conditions of these conditional commitments are the dollar amounts to be paid. The commitment parts include the dollar amounts the owner agents will receive, the goods for the paying agent, and commitments to send the goods from the owner agents.

These conditional commitments have been formed out of proposals which are made in order for the agents to satisfy their commitments of deriving some dollar amount and/or to answer others' requests such as \([O^2ticket@M_M]\perp\) from the musician, \([O^2mp3@W_w]\perp\) and \([O^2book@W_w]\perp\) from the writer. In particular, when a proposal is formed toward another agent, the agent variable \(X\) in the proposal’s rule is assigned to the proposed agent. Once the proposal is accepted, the rule becomes a conditional commitment from the owner agent to the proposed agent. For example, Rule 1 becomes a conditional commitment from the musician to the artist (\(X\) is assigned to \(A\)) and in another case, from the musician to the writer (\(X\) is assigned to \(W\)).

The proposed agents are then responsible to derive the conditions of the conditional commitments. For instance, the artist needs to derive \(O^25\$@A_A\) for the proposal of Rule 1, and \(O^215\$@A_A\) for the proposal of Rule 6 and the writer needs to derive \(O^25\$@W_w\) for the proposal of Rule 1, and \(O^225\$@W_w\) for the proposal of Rule 2.

Whenever these conditions are satisfied, the rules owner agents always fulfill their commitments by making the commitment part available. This is ensured by the linear implication that definitely transforms conditions part to the commitment part. For instance, when the writer makes \(O^25\$@W_w\) available, Rule 2 is applied by the musician to derive \(O^2[25 \square \$@M_M \otimes \square book@W_w \otimes book@W_w]\).

In many cases, the commitment parts of the conditional commitments also include base commitments. For example, there are base commitments of \([O^2book@M_M]\perp\) and \([O^225\$@W_w]\perp\) in Rule 2. These base commitments are added into the rule owner agents’ state formulas. Some of them are fulfilled immediately by available resources, like \([O^2book@M_M]\perp\) being satisfied by \(\square book@M_M\). Others like \([O^225\$@W_w]\perp\) may require further actions from the rule owner agents.

### 8.2.5 Modeling Time

It is important to be able to express things at specific time points, with a duration, and uncertainty in time as well as time relationships among them.

Like many other temporal logics, TLL enables the modeling of specific time points as-
Temporal Linear Logic (TLL) Modeling

associated with an object or event, such as when an action is carried out, or when an event occurs. Specific time points are described by the use of $\Diamond$ operator or multiple copies of it. For example, there are requirements at the second next time point of tickets at the musician ($[\Diamond^2 ticket@M_m]^\perp$), a book at the writer ($[\Diamond^2 book@W_w]^\perp$), and a performance at the artist ($[\Diamond^2 perform@A_a]^\perp$). Based on this ability to specify correct time points for actions or events, time order or sequencing of them can also be captured. For example, the performance by the artist $\Diamond^3 perform@A_a$ is after its purchases of mp3 music and mp3 player, which are at the second next time point.

Moreover, uncertainty in time can represented by the use of $\Box$ and $\Diamond$ and their combinations with $\Diamond$ in TLL modeling. $\Box$ can be used to express the notion of “anytime” or outer non-determinism while $\Diamond$ expresses “sometime” or inner non-determinism.

In the CT example, resources such the book, mp3 music, mp3 player and CD player are available for use at any time, and hence the time of use is unknown but up to the owner’s choice and hence preceded by $\Box$. The actual time points of the availability of those resources are decided by the owner (internally) as appropriate to their uses. For example, as decided by their respective owners, $\Box book@M_m$ becomes $\Diamond^2 book@M_m$, $\Box mp3@M_m$ becomes $\Diamond^2 mp3@M_m$, and $\& i \geq 0 (\Diamond^i ticket@W_w)$ becomes $\Diamond^2 ticket@W_w$. Moreover, a combination of $\Diamond$ and $\Box$ allows us to describe a resource starting available at any time from a time point. For example, after an application of Rule 2 at the second next time point, a book becomes available at any time from that time onwards for the writer: $\Diamond^2 \Box book@W_w$.

However, as all the temporal operators in TLL refer to concrete time points, we cannot express durations in time faithfully. A duration of an event or a resource availability can be expressed by spreading copies of that event or resource over consecutive time points (like $\Diamond A \odot \Diamond^2 A \odot \ldots \Diamond^i A$). In the CT example, tickets belonging to the writer and the artist that are available from now until the third next time point are encoded as $\& i \geq 0 (\Diamond^i ticket@A_a)$ and $\& i \geq 0 (\Diamond^i ticket@W_w)$. One major disadvantage of simulating a duration is that it requires the time range to be provided explicitly.

Similarly, the “until” relationship between two objects or events like an object $A$ exists until an event $B$ occurs can not be specified intuitively. An example of operational modeling of “$A$ until $B$” is $\Box A \otimes \Box (B \otimes A \rightarrow B)$, which requires a provision of the operation to remove $A$ whenever $B$ comes into existence in addition to specifying that $A$ exists anytime. This operational modeling, however, demands extra handling, such as knowing to reserve $\Box A$ and apply the operation $\Box (B \otimes A \rightarrow B)$ whenever $B$ occurs.

Another weakness of TLL modeling with respect to time is that there is a lack of a native
operator between the operator $!$ and $\Box$ (or $\Diamond$). While $\Box A$ and $\Diamond$ allow only one use of $A$ over in infinite time range, $!A$ provides infinite number of $A$ for use any time over infinite time range. It then becomes hard in TLL to express such notion that at a specific time, anytime (or sometime), there are unlimited copies of $A$ that can be used.

Moreover, in TLL, it is difficult to specify things (like formulas $A$, $B$, $C$) with reference to the same time point which is uncertain unless they can all be grouped together like $\Diamond(A \otimes \bigcirc B \otimes C)$. For example, in $\Diamond (A \& \Gamma) \otimes \Diamond \bigcirc B \otimes \Diamond C$, the occurrences of $A$, $B$ and $C$ need not be at the same time point. Moreover, the relationships with temporal operators impose on linear connectives some limitations on their interpretations. For example, in TLL,

$$\bigcirc(A \otimes B) \neq \bigcirc A \otimes \bigcirc B$$

which makes it unusual for such interpretation that having both $A$ and $B$ at the next time point does not imply both having $A$ at the next time point and having $B$ at the next time point. In fact, $\otimes$ should be referred to as the co-existence property and hence it should be said that the co-existence at the next time point of $A$ and $B$ does not imply the co-existence of $A$ at the next time point and $B$ at the next time point. Therefore, care must be taken when associating $\otimes$ (concurrent existence) with the meaning of “and”.

### 8.3 Flexibility in Agent Interaction

We consider several points, including agent interaction and executing specifications, that can increase flexibility.

#### 8.3.1 Flexibility in TLL Modeling

Our use of TLL for modeling agent interactions provides flexibility in several ways.

Firstly, given that flexibility includes the ability to make a sensible choice, having the choices expressed explicitly in the specification of interaction protocols provides agents with an opportunity to reason about them during interaction. In our framework, $\&$, $\oplus$, $\Box$ and $\Diamond$ represent choices of agents. $\&$ and $\oplus$ refer to internal choices and indeterminate possibilities on resources and actions, and $\Box$ and $\Diamond$ refer to internal choices and indeterminate possibilities with respect to time.

In the CT example, many resources like the book, CD player, mp3 player, and music are available at any time point and hence can be used at any time. This is achieved by using $\Box$
in the encoding. Also, the writer has an internal choice that he can choose to listen to music on the mp3 player or on the CD player. The choice is realized by an encoding that uses $\oplus$ in Rule 3 and via the proof construction process which implements the choice.

Secondly, the connective $\otimes$ naturally expresses concurrency, without any ordering constraints. This removes the need (such as in traditional approaches using finite state machines) to impose unnecessary constraints on the order of resources or actions. Hence, flexibility is obtained by allowing the interaction to determine an appropriate ordering. For example, two commitments of the writer in the CT example as described in Section 8.1 are $[\mathcal{O}^2\text{music}@W_w]^\perp$ and $[\mathcal{O}^2\text{book}@W_w]^\perp$, and do not require a particular order to be followed when fulfilling them. Connecting these commitments via $\otimes$ gives the writer the flexibility to work on them in any order.

### 8.3.2 Flexibility from Commitment-based Meanings

Flexibility can also be gained from structuring agent interaction around commitment-based meanings as discussed by [Yolum and Singh, 2002a].

In our approach, protocols are also specified as a set of base commitments and pre-commitments at participating agents. Pre-commitments are utilized to fulfill the agents’ base commitments and from these conditional commitments emerge. Execution of protocols is based on agents fulfilling their commitments. As a result, agents have the flexibility to choose which actions to carry out to satisfy the commitments that have emerged from the protocol’s execution. In comparison to traditional approaches, our approach removes unnecessary constraints on which particular interactive actions to be executed by which agents and on the order among them. In other words, the specification of protocols by using commitments gives agents the flexibility to decide which interactive actions to follow.

### 8.3.3 Flexibility from Negotiation on Pre-commitments

In our framework, conditional commitments emerge from negotiations between agents about pre-commitments. Further flexibility can be gained from the ability to choose a pre-commitment among potentially many proposals from various agents. Hence, agents do not have to follow an externally imposed sequence or a set of conditional commitments, but can flexibly choose suitable ones from the set of pre-commitments via negotiation to satisfy their needs and cope with on-going changes in the environment. This approach also gives agents more autonomy in their utilization of conditional commitments.
8.3.4 Flexibility from Declarative Specifications

Agent interaction in our approach are specified in a declarative manner as a set of base commitments and pre-commitments at participating agents. Hence, rather than determining the details of interactive actions that agents should follow, the approach specifies what should be achieved and what is the norm for engagement (via pre-commitments). The agents then have autonomy over how to use their resources, actions, capabilities and conditional commitments to fulfill their goals and commitments. In particular, as discussed in Chapter 7, agents can make use of proof search techniques to construct proofs of goals and base commitments from resources, actions, capabilities and conditional commitments. From such proofs, appropriate actions can be determined.

Consider the CT example. There are several threads of interaction as follows.

- Sale of a ticket from the writer to the musician
- Sale of a ticket from the artist to the musician
- Sale of a book from the musician to the writer
- Sale of an mp3 player from the writer to the artist
- Sale of mp3 music from the musician to the artist
- Sale of mp3 music from the musician to the writer

The peculiar property of the interaction is that these threads are relatively independent from each other, in that there is no requirement for other threads to be run prior to, during or after any thread. On the one hand, the specification of interaction should not impose any constraints on the order of execution of threads, which is achieved in our approach by the use of the connective $\otimes$.

On the other hand, the order among these threads and how they are triggered should be determined by the agents. In particular, each thread requires from one agent some commitments to derive money for the application of its rule and brings some money to the other agent. These commitments to derive money in turn further cause other threads to occur so that the money (or resource in general) generated from the other threads can fulfill these commitments. The order or chaining of the threads can be determined in a manner similar to matching inputs of one process (thread) to outputs of another. Such matching can be
implemented based on proof search of the corresponding formulas as demonstrated in Section 8.1.2 and exemplified in Section 8.1.3. Moreover, it can be seen from the construction of the interaction that each thread can be invoked by either a request from an agent or by the rule owner agent making a proposal to fulfill its commitments. For example, the request $[\bigcirc^2\text{book}@W_w]_\perp$ may invoke the thread “Sale of a book from the musician” or be initiated by the musician if he has a commitment to obtain some money.

Flexibility is also provided by the ability to deal with unpredictable changes from the environment. In our approach, changes in the environment can be regarded as removing or adding formulas onto the state formulas. Because the proof construction processes take states into account, changes in the state will be reflected in how the proof may be obtained. How the proof is obtained in turn alters the sequences of actions of interacting agents either by introducing or skipping some actions. Hence, agents can flexibly alter their behaviors to deal with changes.

For example, as shown in Section 8.1.4, the agents artist and writer skip the interaction part corresponding to step 5 of the protocol when an mp3 player is brought to the artist during step 4.

Furthermore, flexibility in interaction is evident when protocols can be adjusted and merged by agents. Protocols are specified in terms of a set of base commitments and pre-commitments that concurrently exist among agents. Hence, adding new base commitments (or pre-commitments) or removing existing ones means that the protocol is adjusted, extended or merged with another protocol. The key factor that enables the protocol with such additions or removals to work is the proof construction and negotiation process by agents which will determine how pre-commitments are utilized and combined with agents’ resources, actions and capabilities to fulfill base commitments. Variations amongst base commitments then means variations in the combinations that are required to fulfill them. Also, variations in pre-commitments means variations in the options for proof construction. In addition, a modular approach to protocol specification can further improve the dynamic formation of protocols. Such a modular approach could be based on designing the specification components — base commitments and pre-commitments — to be independent of each other, which simplifies their addition and removal.
8.3.5 Flexibility from Partial Handling

The mechanism to enable partial handling of goals or base commitments and resources or actions has been discussed in Chapter 6. Such partial handling has an important consequence for improving flexibility in agent interaction. Specifically, the division of goals or base commitments and resources or actions can be done dynamically by agents rather than being specified by human designers. More importantly, because the division can be done with respect to an arbitrary sub-formula, it can be done quite flexibly by agents as required by the context or as the agents choose. This means that in an interaction, depending on the context, agents can choose part of a goal or base commitment to fulfill and hence choose a part of the resources or actions available, as appropriate.

8.3.6 Flexibility from Strategies to Deal with Choices and Changes

Chapter 5 discussed how various strategies about choices can be enabled in agent reasoning. As a result agents can choose which strategies to follow in order to fulfill their goals or base commitments. Such strategies include deciding choices in advance or at their associated times, taking a safe approach by preparing for all the possibilities in advance or taking a bold approach by predicting some of the outcomes. Hence, flexibility of interaction can be further gained by enabling various strategies in dealing with choices for agents.

Moreover, as agents apply various strategies to decide on choices to fulfill their goals or base commitments, there is a trade-off between preserving resources or actions reservation and having choices. Consider the two extreme cases below.

The first case is that all the choices are decided. Hence, all the resources or actions and goals or base commitments are known precisely. This means that the resources and actions required will be the minimum.

The other extreme is that all the choices are left undecided. In this case, in order to fulfill goals or base commitments, all possible cases of the goals or base commitments must be prepared. Also, as choices in resources or actions bring uncertainty, more resources or actions are required to make sure that these are sufficient for whatever choices are made. In this case, the resources and actions required will be maximal, as will the agents’ flexibility.

Between the two extremes are those cases where agents decide on some choices and postpone decisions on others. The choices decided will eliminate the need for reservation of the corresponding resources or actions and hence possibly reduce the possible interaction sequences. The choices postponed provide agents with flexibility. Hence, agents can choose
the degree of flexibility they determine is appropriate.

8.4 Timed Petri Nets as Execution Models

Chapters 4 and 7 have addressed a possible execution framework for TLL specifications based on proof search. To justify our choice of this framework, we consider an alternative approach which makes use of the well known formalism of timed Petri nets.

Timed Petri nets (TPNs) have been used as a visual and mathematical tool for modeling concurrent systems. Given that TPNs and TLL have been shown to be strongly related [Hirai, 1999], it is worth investigating further the relationship between TPNs and TLL to take advantage of available techniques for TPNs to address issues of execution and verification of TLL specifications.

In particular, we investigate the mapping between TLL formulas and TPNs based on previous work on Petri nets as models of Intuitionistic linear logic [Engberg and Winskel, 1994; 1993] and the equivalence result between reachability in timed Petri nets and provability in a fragment of TLL in [Hirai, 1999].

8.4.1 Definition of Timed Petri Nets

We consider a description of a (place) timed Petri net (TPN) and related definitions based on [Hirai, 1999] as follows.

**Definition 16.** A (place) **timed Petri net** is a tuple \((Pl, Tr, Ar, \Theta)\), where:

- \(Pl\) is a finite set of places
- \(Tr\) is a finite set of transitions (disjoint from \(Pl\))
- \(Ar\): \((Pl \times Tr) \cup (Tr \times Pl) \rightarrow \mathbb{N}\) is a weighted set of arcs
- \(\Theta\): \(Pl \rightarrow \mathbb{N}\) is a function denoting the waiting time for a given token(s) to become available.

Transitions are defined in Definition 19 below.

A place \(p \in Pl\) can hold an arbitrary non-negative multiplicity of tokens. Each place represents a unique TLL formula type. A token at a place represents a copy of the corresponding formula type. Without being consumed, a token persists in time.

For representation purposes, we denote a token at the place \(p\) as \(\square p\) if \(p\) is available now or \(\circ i \square p\) if \(p\) will become available from exactly the \(i^{th}\) next time point with reference to the starting time of the net as 0. An untimed representation of a token at a place \(p\) in TLL is \(\overline{p}\). For simplicity, unless distinction must be made, we will use \(p\) to mean \(\overline{p}\). Representing
in TLL a token at the place \( p \) that becomes available from exactly the \( i^{th} \) next time point is then straightforward as \( \bigcirc^i \bigtriangleup \bar{p} \).

We further define a set \( Pl_t \) whose elements are of the form \( \bigcirc^i \bigtriangleup p \) for \( p \in Pl \) and \( 0 \leq i \leq n \), where \( n \) is the upper bound in operation time of the net. We assume that the net operates in finite amount of time, and hence \( n \) is finite. This set hence represents all tokens which will become available at the \( i \) time points.

A marking \( M \) of a net is a multiset of tokens. We define a marking at the \( i^{th} \) next time point over the set \( Pl_t \) as follows.

**Definition 17.** A marking at the \( i^{th} \) next time point, denoted as \( \bigcirc^i M_i \), is a multiset with an associate function \( M : Pl_t \rightarrow \mathbb{N} \).

Note that \( \bigcirc^i M_i \) contains all the tokens which will become available exactly at the time \( i \) and are not available before that time. Tokens at a place \( p \) of a marking \( \bigcirc^i M_i \) is denoted as \( \bigcirc^i M_i(p) \). An example is

\[
\bigcirc^2 M_2 = \{ \bigcirc^2 \bigtriangleup p1, \bigcirc^2 \bigtriangleup p1, \bigcirc^2 \bigtriangleup p2 \} \quad \text{and} \quad \bigcirc^4 M_4(p1) = \{ \bigcirc^1 \bigtriangleup p1, \bigcirc^4 \bigtriangleup p1 \}.
\]

The binary operation \( + \) on markings (as multisets) is defined by

\[
(\bigcirc^i M_i + \bigcirc^j M_j)(p) = \bigcirc^i M_i(p) + \bigcirc^j M_j(p), \forall p \in Pl_t.
\]

For example, let \( \bigcirc^2 M_2 = \{ \bigcirc^2 \bigtriangleup p1, \bigcirc^2 \bigtriangleup p2 \} \); \( \bigcirc^2 M_2' = \{ \bigcirc^2 \bigtriangleup p2, \bigcirc^2 \bigtriangleup p3 \} \) then

\[
(\bigcirc^2 M_2 + \bigcirc^2 M_2')(p2) = \{ \bigcirc^2 \bigtriangleup p2, \bigcirc^2 \bigtriangleup p2 \}.
\]

Hence, for all places \( p \in Pl_t \),

\[
\bigcirc^2 M_2 + \bigcirc^2 M_2' = \{ \bigcirc^2 \bigtriangleup p1, \bigcirc^2 \bigtriangleup p2, \bigcirc^2 \bigtriangleup p2, \bigcirc^2 \bigtriangleup p3 \}.
\]

In TLL, a multiset represents a multiplicative conjunction of formulas. Hence we will write the binary operation \( + \) on multisets as the connective \( \otimes \).

We denote \( \overline{\bigcirc^i M_i} \) as the TLL formula corresponding to the marking \( \bigcirc^i M_i \). Given \( \bigcirc^1 M_1 = \{ \bigcirc^1 \bigtriangleup p1, \bigcirc^1 \bigtriangleup p1, \bigcirc^1 \bigtriangleup p2 \} \), for example, then \( \overline{\bigcirc^3 M_3} = \bigcirc^1 \bigtriangleup p1 \otimes \bigcirc^1 \bigtriangleup p1 \otimes \bigcirc^1 \bigtriangleup p2 \). Also, the correspondence between the multiset binary operation \( + \) and \( \otimes \) yields

\[
\overline{\bigcirc^i M_i + \bigcirc^j M_j} = \overline{\bigcirc^i M_i} \otimes \overline{\bigcirc^j M_j}.
\]

During operation period of the net, as time passes, the number of time points after which tokens of a marking will become available is reduced. For example, after one time point passing, a token \( \bigcirc^i \bigtriangleup p \) becomes \( \bigcirc^{i-1} \bigtriangleup p \). We denote a marking \( \bigcirc^i M_i \) as \( \bigcirc^{i-1} M_i \) after one time point passes and similarly as \( \bigcirc^{i-2} M_i \) after two time points pass.
Definition 18. A state $S$ of a net is a multiset of markings in time and is defined as

$$S = M_0 + \bigcirc M_1 + \bigcirc^2 M_2 + \cdots + \bigcirc^n M_n = \bigcup_{i \geq 0} \bigcirc^i M_i,$$

where $\bigcirc^i M_i$ is a marking at the $i^{th}$ next time points and $n$ is the upper bound in operation time of the net.

From the definition, a state contains all the markings associated with all the time points of the operation period. Given that each marking $\bigcirc^i M_i$ represents all the tokens that (will) become available at a specific time point, a state contains all the tokens that are currently available and will be available during the operation period of the net. In other words, a state is a distribution of tokens over places of the net in time.

The TLL formula corresponding to a state $S$ is:

$$\bar{S} = \bar{M}_0 + \bigcirc \bar{M}_1 + \bigcirc^2 \bar{M}_2 + \cdots + \bigcirc^n \bar{M}_n = \bar{M}_0 \otimes \bigcirc \bar{M}_1 \otimes \cdots \otimes \bigcirc^n \bar{M}_n.$$

The + operation among states is then defined as $(S + S')(a) = S(a) + S'(a), \forall a \in Pl_t$. In the notation of multisets, + is defined as follows:

$$S + S' = M_0 + M'_0 + \bigcirc M_1 + \bigcirc M'_1 + \cdots + \bigcirc^i M_i + \bigcirc^i M'_i + \cdots = \sum_i (\bigcirc^i M_i + \bigcirc^i M'_i).$$

The corresponding TLL formula is:

$$\bar{S} + \bar{S}' = \bar{M}_0 \otimes \bar{M}'_0 \otimes \bigcirc \bar{M}_1 \otimes \bigcirc \bar{M}'_1 \otimes \cdots \otimes \bigcirc^i \bar{M}_i \otimes \bigcirc^i \bar{M}'_i \otimes \cdots.$$

Derivations of states can be of two forms: derivations due to firing transitions and time derivations.

Definition 19. Let $\bigcirc^i \tau^n_0$ be a multiset over $Pl_t$ containing tokens which will become available at the $i^{th}$ next time point. A firing transition $\tau \in Tr$ of a (place) timed Petri net $N$ is a mapping which takes a multiset of \((-\tau_0)\) and gives another multiset $(\tau^n_0 + \bigcirc \tau^+_1 + \bigcirc^2 \tau^+_2 + \cdots + \bigcirc^n \tau^+_n )$.

A firing derivation is enabled at $S$ if and only if $-\tau \subseteq M_0$ or equivalently $M_0 = M_{0r} + -\tau$. If a firing is enabled at an instant, it is fired at that instant and will consume $-\tau$.

Denote $S[\delta]$ as a state reached by $S$ via a firing derivation $\tau$. Then from the definition,

$$S[\tau] = [M_{0r} + \bigcirc M_1 + \bigcirc^2 M_2 + \cdots] + [\tau_0^+ + \bigcirc \tau^+_1 + \bigcirc^2 \tau^+_2 + \cdots]$$

$$= M_{0r} + \tau_0^+ + \bigcirc M_1 + \bigcirc \tau^+_1 + \bigcirc^2 M_2 + \bigcirc^2 \tau^+_2 + \cdots.$$
The TLL representation of a transition $\tau$ is

$$\tau = \tau_0^{-} \rightarrow \tau_0^{+} \otimes \circ \tau_1^{+} \otimes \circ \tau_2^{+} \otimes \ldots \otimes \circ n \tau_n^{+}.$$ 

Denote a state $S$ in TLL as

$$\mathcal{S} = \mathcal{M}_0 \otimes \circ \mathcal{M}_1 \otimes \circ 2 \mathcal{M}_2 \otimes \ldots$$

In TLL notation, a firing derivation is expressed as

$$\mathcal{S} \otimes \tau = [\mathcal{M}_0 \otimes \circ \tau \otimes \circ \mathcal{M}_1 \otimes \ldots] \otimes [\tau^{-} \rightarrow \tau_0^{+} \otimes \circ \tau_1^{+} \otimes \ldots]$$

$$= \mathcal{M}_0 \otimes \tau_0^{+} \otimes \circ \mathcal{M}_1 \otimes \circ \tau_1^{+} \otimes \ldots$$

$$= \mathcal{S}[\tau].$$

**Definition 20.** Let $S = \mathcal{M}_0 + \circ \mathcal{M}_1 + \circ 2 \mathcal{M}_2 + \ldots$ be a state in a (place) timed Petri Net $N$. A **time derivation** $\delta$ is defined as having a reached state $S[\delta]$ at the next time point as follows:

$$S[\delta] = \mathcal{M}_0 + \mathcal{M}_1 + \circ \mathcal{M}_2 + \circ 2 \mathcal{M}_3 + \ldots.$$ 

In TLL notation, $S[\delta] = \mathcal{M}_0 \otimes \circ \mathcal{M}_1 \otimes \circ \mathcal{M}_2 \otimes \circ 2 \mathcal{M}_3 \otimes \ldots$. Note that $S[\delta] \not= \circ \mathcal{S}$ as

$$\circ [\mathcal{M}_0 \otimes \circ \mathcal{M}_1 \otimes \circ \mathcal{M}_2 \otimes \ldots] \not= [\mathcal{M}_0 \otimes \circ \mathcal{M}_1 \otimes \circ 2 \mathcal{M}_2 \otimes \ldots].$$

We denote that a state $S'$ is reached from a state $S$ by a sequence of firing derivations and time derivations over a period of $t$ time units as $S \xrightarrow{t} S'$. Note that, the notation $\xrightarrow{t}$ does not impose any particular order in time of the firing derivations.

An example is $S = \{\square p_1, \square p_2\}$, which means the state has a token at place $p_1$ and a token at place $p_2$ at the initial time.

A transition $\tau$ is given in which

$$\tau = \{p_1\}, \quad \tau_0^{-} = \{\square p_1\}, \quad \circ \tau_1^{+} = \{\circ \square p_2, \circ \square p_2\}, \quad \circ 2 \tau_2^{+} = \emptyset, \ldots.$$ 

Hence, the transition $\tau$ is enabled at the time $t = 0$ and it is fired. A state $S'$ is reached. $S' = \{\square p_1, \circ \square p_2, \circ \square p_2\}$. The time derivation of $S'$ at the time $t+1$ is $S'' = S'[\delta] = \{\square p_1, \square p_2, \square p_2\}.$
8.4.2 Properties of (Place) Timed Petri Nets

In order to provide the analysis of (place) timed Petri nets as models of a fragment of TLL (see Section 8.4.3), we formalize some properties of (place) timed Petri nets as follows. For consistency, we also re-establish the equivalence between reachability in (place) timed Petri nets and provability in TLL, which was established in [Hirai, 1999] in a different manner.

**Lemma 8.4.1.** Let $O^i M_i$ be a marking in a place timed Petri Net as defined above at the $i^{th}$ next time point. Let $O^i M_i$ be $O^i \Box p_1 + O^i \Box p_2 + \cdots + O^i \Box p_j + \ldots$.

Let $O^{i-1} M_i$ be the result of replacing every token in $O^i M_i$ by a token at the same place but at the $(i - 1)^{th}$ next time point, so that $O^{i-1} M_i = O^{i-1} \Box p_1 + O^{i-1} \Box p_2 + \cdots + O^{i-1} \Box p_j + \ldots$.

Then we have $O^{i} M_i \vdash O^{i-1} M_i$.

**Proof:** We make use of the following result in TLL $O A \otimes O B \vdash O (A \otimes B)$.

\[
\frac{A \vdash A \quad B \vdash B}{A, B \vdash A \otimes B} \otimes L
\]

\[
\frac{O A, O B \vdash O (A \otimes B)}{O A \otimes O B \vdash O (A \otimes B)} \otimes L
\]

By applying this result repeatedly, we can prove

$O A \otimes O B \otimes \ldots \vdash O (A \otimes B \otimes \ldots)$.

The TLL formula of $O^i M_i$ is $O^{i} M_i = O^i \Box p_1 \otimes O^i \Box p_2 \otimes \ldots \otimes O^i \Box p_j \ldots$.

Also, $O^{i-1} M_i = O^{i-1} \Box p_1 \otimes O^{i-1} \Box p_2 \otimes \ldots \otimes O^{i-1} \Box p_j \ldots$.

We have

\[
\ldots \frac{O^{i} \Box p_1 \otimes O^{i} \Box p_2 \otimes \ldots \otimes O^{i} \Box p_j \ldots \vdash O(O^{i-1} \Box p_1 \otimes O^{i-1} \Box p_2 \otimes \ldots \otimes O^{i-1} \Box p_j \ldots)}{O^{i} M_i \vdash O^{i-1} M_i}
\]

**Theorem 8.4.2.** Let $S$ be a state of a place timed Petri net as defined above. Let $S[\delta]$ be a state reached from $S$ after a time derivation. Then

$S \vdash S[\delta]$

**Proof:**

Denote the formula corresponding to the state $S$ as

$S = M_0 \otimes O M_1 \otimes O^2 M_2 \otimes \ldots$
Then $S[\delta] = M_0 \otimes M_1 \otimes \bigcirc M_2 \otimes \ldots$.

As $M_0$ contains tokens of the form $\square p_i$, $M_0$ can be regarded as formula of the form $\square \Pi$. Hence, $M_0$ can act as a context formula (unchanged) in the $\bigcirc$ rule. We have the following proof:

\[
\begin{align*}
M_0 \otimes M_1 \otimes \bigcirc M_2 \otimes \ldots & \vdash M_0 \otimes M_1 \otimes \bigcirc M_2 \otimes \ldots \\
M_0 & \vdash \bigcirc S[\delta]
\end{align*}
\]

From Lemma 8.4.1, we have $\bigcirc i M_i \vdash \bigcirc \bigcirc^{-1} M_i$. Hence, $\bigcirc M_1 \vdash \bigcirc \bigcirc^2 M_2 \vdash \bigcirc \bigcirc M_2$, and so on.

\[
\begin{align*}
M_0 & \vdash M_0 \\
\bigcirc M_1 & \vdash \bigcirc M_1 \\
\bigcirc^2 M_2 & \vdash \bigcirc \bigcirc M_2 \\
M_0 \otimes \bigcirc M_1 \otimes \bigcirc^2 M_2 \otimes \ldots & \vdash M_0 \otimes \bigcirc M_1 \otimes \bigcirc \bigcirc M_2 \otimes \ldots \\
S & \vdash M_0 \otimes \bigcirc M_1 \otimes \bigcirc \bigcirc M_2 \otimes \ldots
\end{align*}
\]

Hence, by the cut rule, we have a proof of the lemma

\[
S \vdash [M_0 \otimes \bigcirc M_1 \otimes \bigcirc \bigcirc M_2 \otimes \ldots] [M_0 \otimes \bigcirc M_1 \otimes \bigcirc \bigcirc M_2 \otimes \ldots] \vdash \bigcirc S[\delta]
\]

\[
S \vdash \bigcirc S[\delta]
\]

\[\blacksquare\]

**Lemma 8.4.3.** Let $S$ and $S''$ be states of a place timed Petri net as defined above. Let $S'$ be a state reached from $S$ after firing derivations and time derivations over a period of $t$ time unit(s). Let $S''[\delta]^t$ denote the state $S''$ after undergoing $t$ unit time derivations. We have

\[
S \xrightarrow{t} S' \implies S + S'' \xrightarrow{t} S' + S''[\delta]^t
\]

**Proof:** Suppose that $S$ reaches $S'$ in $n$ derivations, of which $t$ derivations are time derivations. We proceed by induction on the length of the derivations.

**Base case:** When $n = 0$, $t = 0$, we have $S \rightarrow S$, $S''[\delta]^0 = S''$, and so $S + S'' \rightarrow S + S''[\delta]^0$.

**Hypothesis:** The Lemma holds for the case of $n - 1 \geq 0$ derivations in $t_{n-1} \geq 0$ time units:

\[
S \xrightarrow{t_{n-1}} S'_{n-1} \implies S + S'' \xrightarrow{t_{n-1}} S'_{n-1} + S''[\delta]^{t_{n-1}}.
\]

**Induction step:** we need to prove for the case of $n$ derivations:

\[
S \xrightarrow{t_n} S'_n \implies S + S'' \xrightarrow{t_n} S'_n + S''[\delta]^{t_n}.
\]
Let $S'_{n-1} = M_0 + \bigcirc M_1 + \bigcirc^2 M_2 + \ldots$, and $S''(\delta)^{t_{n-1}} = M''_0 + \bigcirc M''_1 + \bigcirc^2 M''_2 + \ldots$

The state reached after $n - 1$ transitions $S'_{n-1} + S''(\delta)^{t_{n-1}}$ is $= (M_0 + M''_0) + (\bigcirc M_1 + \bigcirc M''_1) + (\bigcirc^2 M_2 + \bigcirc^2 M''_2) + \ldots$.

After $n - 1$ transitions, there are two cases of the next transition, either a firing derivation or a time derivation.

In the case when the next transition is a firing derivation $\tau$, we have

\[
\tau = \overline{\tau}_0 \Rightarrow \overline{\tau}_0^+ \otimes \overline{\tau}_1^+ \otimes \overline{\tau}_2^+ \otimes \ldots \otimes \overline{\tau}_n^+
\]

then from (*), the state reached after $n$ transitions from $S + S''$ is

\[
[S'_{n-1} + S''(\delta)^{t_{n-1}}]|\tau\rangle = (M_0 - \tau + \tau_0^+ + M''_0) + (\bigcirc M_1 + \bigcirc M''_1 + \bigcirc \tau_1^+) + \ldots.
\]

Whereas $S'_n = S'_{n-1}[\tau\rangle = (M_0 - \tau + \tau_0^+) + (\bigcirc M_1 + \bigcirc \tau_1^+) + \ldots$.

Also, because the transition is $\tau$, there is no change in time, $t_n = t_{n-1}$, and so $S''(\delta)^{t_n} = S''(\delta)^{t_{n-1}}$.

Hence, together, $S'_n + S''(\delta)^{t_n}$

\[
= [(M_0 - \tau + \tau_0^+) + (\bigcirc M_1 + \bigcirc \tau_1^+) + \ldots] + [M''_0 + \bigcirc M''_1 + \ldots]
\]

\[
= (M_0 - \tau + \tau_0^+ + M''_0) + (\bigcirc M_1 + \bigcirc M''_1 + \bigcirc \tau_1^+) + \ldots.
\]

Therefore, $[S'_{n-1} + S''(\delta)^{t_{n-1}}]|\tau\rangle = S'_n + S''(\delta)^{t_n}$.

In the case when the next transition is a time derivation $\delta$ of one time unit, we have from (*) that the state reached after $n$ transitions is

\[
[S'_{n-1} + S''(\delta)^{t_{n-1}}]|\delta\rangle = (M_0 + M''_0 + M_1 + M''_1) + (\bigcirc M_2 + \bigcirc M''_2) + (\bigcirc^2 M_3 + \bigcirc^2 M''_3) + \ldots.
\]

Whereas $S'_n = S_{n-1}[\delta\rangle = M_0 + M_1 + \bigcirc M_2 + \bigcirc^2 M_3 + \ldots$.

Also, $S''(\delta)^{t_n} = S''(\delta)^{t_{n-1}}[\delta\rangle = M''_0 + M''_1 + M''_2 + \bigcirc^2 M''_3 + \ldots$.

Then together, $S'_n + S''(\delta)^{t_n}$

\[
= [M_0 + M_1 + \bigcirc M_2 + \bigcirc^2 M_3 + \ldots] + [M''_0 + M''_1 + \bigcirc^2 M''_2 + \bigcirc^2 M''_3 + \ldots]
\]

\[
= (M_0 + M''_0 + M_1 + M''_1) + (\bigcirc M_2 + \bigcirc M''_2) + (\bigcirc^2 M_3 + \bigcirc^2 M''_3) + \ldots.
\]

Therefore, $[S'_{n-1} + S''(\delta)^{t_{n-1}}]|\delta\rangle = S'_n + S''(\delta)^{t_n}$.
From the hypothesis and for all the cases of the next transition, we have
\[ S + S'' \xrightarrow{t} S'_n + S''[\delta]^t. \]

By induction, we can conclude that
\[ S \xrightarrow{t} S' \implies S + S'' \xrightarrow{t} S'_n + S''[\delta]^t. \]

We consider further a proof system for Intuitionistic TLL discussed in [Hirai, 1999].

**Definition 21.** Let \( \text{ITLL}^0 \) be a subsystem of Intuitionistic TLL that is obtained by replacing the sequent rule
\[
\begin{array}{c}
\Gamma, \Delta \vdash G \\
\hline
\Gamma \vdash F \\
\end{array}
\] \( \otimes R \)

by the sequent rule
\[
\begin{array}{c}
\Gamma, F \vdash F, \Delta, G \vdash G \\
\hline
\Gamma, \Delta, F, G \vdash F \otimes G \otimes R^0 \\
\end{array}
\]

and in which all atoms are of the form \( \square p \).

Also, we will write a sequent of the form \( 1 \otimes \overline{S} \vdash \circ \overline{1}(1 \otimes \overline{S}) \) as \( \overline{S} \vdash \circ \overline{1} \overline{S} \). Sequents of the form \( \overline{S} \vdash \circ \overline{1} \overline{S} \) are understood in this way unless otherwise stated.

We also equip the proof system \( \text{ITLL}^0 \) with a set of axioms that are based on the set of transitions \( Tr \) of the net \( N \). In particular, for each transition in \( Tr \) (in TLL notation)
\[
\overline{\tau} = \tau_0^+ \otimes \circ \tau_1^+ \otimes \circ \circ \tau_2^+ \otimes \ldots \otimes \circ^n \tau_n^+, \\
\]
the following axiom is added:
\[
\vdash \tau_0^+ \otimes \circ \tau_1^+ \otimes \circ \circ \tau_2^+ \otimes \ldots \otimes \circ^n \tau_n^+. \\
\]

In this proof system, a correspondence between reachability in a (place) timed Petri net and provability of the corresponding state formulas in the logic fragment can be established [Hirai, 1999].

**Theorem 8.4.4.** Let \( N \) be a (place) timed Petri net. Let \( S, S' \) be states of it and \( Tr \) be a set of its transitions. Let \( Ax \) be a set of axioms corresponding to \( Tr \). Then \( S' \) is reachable from \( S \) after a passage of \( t \) time units if and only if the sequent \( \overline{S} \vdash \circ \overline{1} \overline{S}' \) is provable in \( \text{ITLL}^0 \) using \( Ax \).
**Proof:** a proof can be found in [Hirai, 1999].

We consider a special case of the theorem.

**Corollary 8.4.5.** \( S \xrightarrow{0} S' \iff \overline{S} \vdash \overline{S'} \).

Moreover, we have following lemma regarding derivations of a combination of states.

**Lemma 8.4.6.** \( (S + S')[\delta]^t = S[\delta]^t + S'[\delta]^t \)

**Proof.** \( S + S' = M_0 + M_0' + \bigcirc M_1 + \bigcirc M_1' + \ldots \)

\( (S + S')[\delta]^t = (M_0 + M_0' + M_1 + M_1' + \cdots + M_t + M_t') + \bigcirc M_{t+1} + \bigcirc M_{t+1}' + \ldots \)

\( S[\delta]^t = (M_0 + M_1 + \cdots + M_t) + \bigcirc M_{t+1} + \ldots \)

\( S'[\delta]^t = (M_0' + M_1' + \cdots + M_t') + \bigcirc M_{t+1}' + \ldots \)

Hence, \( S[\delta]^t + S'[\delta]^t = (M_0 + M_0' + M_1 + M_1' + \cdots + M_t + M_t') + \bigcirc M_{t+1} + \bigcirc M_{t+1}' + \cdots = (S + S')[\delta]^t \) \( \square \)

**8.4.3 Timed Petri Nets as Models of a Fragment of TLL**

Our approach extends the approach in [Engberg and Winskel, 1993; 1994] on Petri Nets as Models of Linear Logic to cover the relationships between Temporal Linear Logic and timed Petri nets.

Firstly, we define a fragment of Intuitionistic TLL that is suitable for mapping.

Consider a sub system **PITLL** of Intuitionistic TLL (ITLL). Its language is defined as:

\[
A ::= \top | \bot | \square a | \bigcirc A | A \otimes A \rightarrow A | A \& A
\]

where \( a \) ranges over linear logic atoms.

Note that the symbol \( \square \) is only used in atomic propositions. This together with a lack of the \( \oplus \) connective makes this fragment fundamentally different from MCA (definition 9).

The sequent calculus corresponding to this fragment clearly can exclude the left and right introduction rules for \( \oplus \), and the modal rules for left and right \( \square \).

**Interpretation on Timed Petri Nets**

In the fragment, all atomic TLL formulas are of the form \( \square a \), where \( a \) is a proposition. \( \square a \) is interpreted as a singleton multiset denoted as \( \square a \). An assertion \( A \) (TLL formula) is considered at the present time. To consider an assertion \( A \) in the future, we use \( \bigcirc^t A \).

Similar to the approach in [Engberg and Winskel, 1993; 1994], we make use of the notion that denotation of an assertion (TLL formula) is a set of requirements (states) sufficient to
establish it. Specifically, an assertion is denoted as a subset of states at the present time which is downward closed with respect to the reachability of the net with no time derivations. Here, a downward closure of a state S is defined as:

\[ \downarrow S = \{ S' \in \mathcal{S} | S' \xrightarrow{0} S \} \]

where \( \mathcal{S} \) is the set of all the states at present time.

Formally, we establish an interpretation of assertions in TLL as follows.

**Definition 22.** Let \( N \) denote a (place) timed Petri net \( N \) as defined in Section 8.4.1. Let \( \mathcal{S} \) be all the states at the present, and \( 0 \) be the state corresponding to the empty multi-set.

Then formulas in the fragment \( \text{PITLL} \) are interpreted in \( N \) as follows.

\[
\begin{align*}
[\top]_N &= \mathcal{S} \\
[\bot]_N &= \emptyset \\
[1]_N &= \{ S | S \xrightarrow{0} 0 \} \\
[\square a]_N &= \{ S | S \xrightarrow{0} a \} \\
[\square \! A]_N &= \{ S | S[\delta] \in [[A]] \} \\
[A \otimes B]_N &= \{ S | \exists S_A \in [[A]] , \exists S_B \in [[B]] , S \xrightarrow{0} S_A + S_B \} \\
[A \rightarrow B]_N &= \{ S | \forall S_A \in [[A]] , S + S_A \in [[B]] \} \\
[A \& B]_N &= [[A]] \cap [[B]].
\end{align*}
\]

With respect to this interpretation, we will establish that a timed Petri net \( N \) is a model of the fragment of TLL mentioned above by proving that if \( A \vdash B \) then \([A]_N \) semantically entails \([B]_N \) and vice versa.

Given that set inclusion corresponds to implication, a semantic entailment between assertions \( A \) and \( B \) is given as: \( A \vdash N B \iff [[A]]_N \subseteq [[B]]_N \)

Firstly, we consider soundness. Note that from now on, we write \([A] \) for a shorthand and assume the reference to the net \( N \) is understood in the context.

**Theorem 8.4.7.** (Soundness) If \( \Gamma \vdash A \) then \([\Gamma] \subseteq [[A]] \)

**Proof:** We need to prove that each rule is sound under this interpretation.

**Case:**

\[ A \vdash A \]

It is clear that \([A] \subseteq [[A]] \).

**Case:**

\[ \Gamma \vdash A \quad \Delta \vdash B \]

\[ \Gamma, \Delta \vdash A \otimes B \]
From $S \in [[\Gamma \otimes \Delta]] \Rightarrow \exists S^*_\Gamma \in [[\Gamma]], \exists S^*_\Delta \in [[\Delta]], S \rightarrow S^*_\Gamma + S^*_\Delta$.

Also, from $\Gamma \vdash A \Rightarrow \forall S_{\Gamma} \in [[\Gamma]], S_{\Gamma} \in [[A]]$. Given $S^*_\Gamma \in [[\Gamma]]$, then $S^*_\Gamma \in [[A]]$.

From $\Delta \vdash B \Rightarrow \forall S_{\Delta} \in [[\Delta]], S_{\Delta} \in [[B]]$. Given $S^*_\Delta \in [[\Delta]]$, then $S^*_\Delta \in [[B]]$.

Hence, $S_{\Gamma}^* \in [[A]]$ and $S_{\Delta}^* \in [[B]]$. From the definition, we have $S_{\Gamma}^* + S_{\Delta}^* \in [[A \otimes B]]$.

From $S \rightarrow S_{\Gamma}^* + S_{\Delta}^* \in [[A \otimes B]]$, then $S \in [[A \otimes B]]$.

**Case:**

\[
\begin{array}{c}
\Gamma, A \vdash C \\
\hline
\Gamma, A \& B \vdash C
\end{array}
\]

We have $S \in [[\Gamma \otimes (A \& B)]] \iff \exists S_{\Gamma} \in [[\Gamma]], \exists S_{AB} \in [[A]] \cap [[B]], S \rightarrow S_{\Gamma} + S_{AB}$.

As $S_{AB} \in [[A]] \cap [[B]] \Rightarrow S_{AB} \in [[A]]$, we hence denote $S_A = S_{AB}$.

Hence, $S \rightarrow S_{\Gamma} + S_{AB}$ also means $S \rightarrow S_{\Gamma} + S_A$. From the definition, $S_{\Gamma} + S_A \in [[\Gamma \otimes A]]$, we then have $S \in [[\Gamma \otimes A]]$. In order to conclude $\Gamma, A \& B \vdash C$, we then need to establish that $S \in [[C]]$.

In fact, the antecedent gives

$\Gamma, A \vdash C \iff \forall S', S' \in [[\Gamma \otimes A]] \Rightarrow S' \in [[C]]$.

As from above $S \in [[\Gamma \otimes A]]$, let $S' = S \in [[\Gamma \otimes A]]$ then we have $S \in [[C]]$.

**Case:**

\[
\begin{array}{c}
\Gamma \vdash A, \Gamma \vdash B \\
\hline
\Gamma \vdash A \& B
\end{array}
\]

For any $S \in [[\Gamma]]$, from the antecedents, $\Gamma \vdash A$ and $\Gamma \vdash B$, we have $S \in [[A]]$ and $S \in [[B]]$.

Hence $S \in [[A]] \cap [[B]] \iff S \in [[A \& B]]$. In other words, $\forall S \in [[\Gamma]], S \in [[A \& B]] \iff \Gamma \vdash A \& B$.

**Case:**

\[
\begin{array}{c}
\Gamma \vdash A, \Delta, B \vdash C \\
\hline
\Gamma, \Delta, A \rightarrow B \vdash C
\end{array}
\]

Consider $S \in [[\Gamma \otimes \Delta \otimes (A \rightarrow B)]]$

$\iff \exists S_{\Gamma} \in [[\Gamma]], \exists S_{\Delta} \in [[\Delta]], \exists S_{A \rightarrow B} \in [[A \rightarrow B]], S \rightarrow S_{\Gamma} + S_{\Delta} + S_{A \rightarrow B}$.

We need to prove that $S \in [[C]]$.

$S_{A \rightarrow B} \in [[A \rightarrow B]] \iff \forall S_A \in [[A]], S_A + S_{A \rightarrow B} \rightarrow S_B \in [[B]]$.

From the antecedent $\Gamma \vdash A$, we have $\forall S, S_{\Gamma} \in [[\Gamma]] \Rightarrow S_{\Gamma} \in [[A]]$. Hence $S_{\Gamma} + S_{A \rightarrow B} \rightarrow S_B \in [[B]]$. 


We then have \( S^0 \rightarrow S_\Gamma^0 + S_\Delta^0 + S_{A \rightarrow B}^0 \rightarrow S_\Delta^0 + S_B^0 \) from Lemma 8.4.3, which means that \( S \in [[\Delta \otimes B]] \).

From the antecedent \( \Delta, B \vdash C \), we have \( \forall S', S' \in [[\Delta \otimes B]] \Rightarrow S' \in [[C]] \). Let \( S' = S \) then we have \( S \in [[C]] \).

Hence \( S^0 \rightarrow S_\Delta + S_B \Rightarrow S \in [[C]] \).

**Case:**

\[
\frac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B}
\]

We have to prove that for any \( S \in [[\Gamma]] \Rightarrow S \in [[A \Rightarrow B]] \)
\[\Leftrightarrow \forall S \in [[\Gamma]], \forall S_A \in [[A]], S_A + S \in [[B]]\] .

We have that \( \forall S \in [[\Gamma]], \forall S_A \in [[A]], S + S_A \in [[\Gamma \otimes A]] \) from the definition.

From the antecedent \( \Gamma, A \vdash B \), then \( \forall S' \in [[\Gamma \otimes A]], S' \in [[B]] \). Let \( S' = S_A + S \), then also \( S_A + S \in [[B]] \).

**Case:**

\[
\frac{\Gamma, \Xi \vdash A}{\Gamma, \Xi \vdash B}
\]

Note that \( \Gamma \) here contains token formulas all of which are of the form \( \Box p_i \). In other words, \( \Gamma \) is of the form \( \Box \Psi \). Because these tokens are unchanged as time passes, we have that \( \forall S_\Gamma \in [[\Gamma]], S_\Gamma[\delta] \in [[\Gamma]] \).

Consider \( \forall S \in [\Gamma \otimes \Xi] \Leftrightarrow \forall S, \exists S_\Gamma \in [[\Gamma]], \exists S_\Xi[\delta] \in [[\Xi]], S^0 \rightarrow S_\Gamma + S_\Xi[\delta] \). We need to prove that \( S \in [[\Gamma \otimes A]] \). Given that \( S_\Gamma[\delta] \in [[\Gamma]] \) as above, \( S_\Gamma[\delta] + S_\Xi[\delta] \in [[\Gamma \otimes \Xi]] \).

From the antecedent, \( \Gamma, \Xi \vdash A \Rightarrow \forall S' \in [[\Gamma \otimes \Xi]], S' \in [[A]] \).

Taking \( S' = S_\Gamma[\delta] + S_\Xi[\delta] \), then \( S_\Gamma[\delta] + S_\Xi[\delta] \in [[A]] \)
\[\Leftrightarrow [S_\Gamma + S_\Xi][\delta] \in [[A]] \], according to Lemma 8.4.6
\[\Leftrightarrow [S_\Gamma + S_\Xi] \in [[\Gamma \otimes A]] \].

As \( S^0 \rightarrow S_\Gamma + S_\Xi \), and \( [S_\Gamma + S_\Xi] \in [[\Gamma \otimes A]] \), then \( S \in [[\Gamma \otimes A]] \).

**Case:**

\[
\frac{\Gamma \vdash C}{\Gamma, \Gamma \vdash C}
\]

\( \forall S \in [[\Gamma \otimes 1]] \Leftrightarrow \forall S, \exists S_\Gamma \in [[\Gamma]], \exists S_1 \in [[1]], S^0 \rightarrow S_\Gamma + S_1. \)
We need to prove $S \in [[C]]$. Note that $\forall S_1 \in [[1]], S_1 \overrightarrow{0} 0$. Hence, $S \overrightarrow{0} S_\Gamma + S_1 \overrightarrow{0} S_\Gamma + 0 = S_\Gamma$ from Lemma 8.4.3.

The antecedent $\Gamma \vdash C \Rightarrow \forall S_\Gamma' \in [[\Gamma]], S_\Gamma' \in [[C]]$. Let $S_\Gamma' = S_\Gamma$ then $S_\Gamma \in [[C]]$. As $S \overrightarrow{0} S_\Gamma$, $S \in [[C]]$.

Case:

\[ \Gamma \vdash \top \]

This is obvious as an empty multiset $0 \in [[1]]$.

Case:

\[ \Gamma \vdash \bot \]

$\forall S \in [[\Gamma]], S \in [[\top]]$ as $[[\top]]$ is the set of all states. 

In order to prove the completeness theorem, we first prove the following lemma.

Lemma 8.4.8. $[[\Gamma]] = \{S|S \vdash \Gamma\}$

Proof: We proceed by induction on the structure of $\Gamma$.

Base cases: We consider in turn the cases for $\Gamma$ as $\top, \bot, 1$ and $\Box a$.

Case: $[[\top]] = \{S|S \vdash \top\}$.

$[[\top]] = \mathcal{P}$, which contains all the states at present time. We also have $\forall S \in \mathcal{P}, S \vdash \top$.

Case: $[[\bot]] = \{S|S \vdash \bot\}$.

Because there is no state such that $S \vdash \bot$, $\{S|S \vdash \bot\} = \emptyset = [[\bot]]$.

Case: $[[1]] = \{S|S \vdash 1\}$.

From the definition, $[[1]] = \{S|S \overrightarrow{0} \emptyset\}$. As $S$ reaches a state corresponds to an empty multiset $\emptyset$, $S$ must be empty. Hence, $S \vdash 1$ corresponds to $\vdash 1$, which is an axiom in TLL.

Case: $[[\Box a]] = \{S|S \vdash \Box a\}$.

$[[\Box a]] = \{S|S \overrightarrow{0} \Box a\}$, from the definition $\iff \{S|S \vdash \Box a\}$, from Corollary 8.4.5.

Induction step

Hence we have that the lemma holds when $\Gamma$ contains the basic forms above, i.e. that $[[\Gamma]] = \{S|S \in [[\Gamma]]\} = \{S|S \vdash \Gamma\}$, for $\Gamma \in \{\top, \bot, 1, \Box a\}$. 

We now proceed to the other cases for $\Gamma$.

**Case:** $[[\bigcirc A]] = \{ S[\overline{S} \vdash \bigcirc A] \}$.  
We first prove that if $S \in [[\bigcirc A]]$ then $\overline{S} \vdash \bigcirc A$. 
$[[\bigcirc A]] = \{ S[S[\delta] \in [[A]]] \}$, from the definition.

$\iff \{ S[S\exists S_A \in [[A]], S[\delta] \xrightarrow{0} S_A] \}$, from the definition of downward closure.

$\Rightarrow \{ S[S\exists S_A \in [[A]], S \xrightarrow{1} S_A] \}$, as $S[\delta] \xrightarrow{0} S_A$ means $S \xrightarrow{1} S_A$.

$\iff \{ S[S\exists S_A \in [[A]], \overline{S} \vdash \overline{S_A}] \}$, from theorem 8.4.4.

$\iff \{ S[S_S \xrightarrow{A}; \overline{S} \vdash \overline{S_A}] \}$, from induction steps above.

$\Rightarrow \{ S[S \circ S_S \xrightarrow{A}; \overline{S} \vdash \overline{S_A}] \}$, applying $\circ$ rule.

$\Rightarrow \{ S[S \overline{S} \vdash \bigcirc A] \}$, applying cut rule.

Next we prove that if $\overline{S} \vdash \bigcirc A$ then $S \in [[\bigcirc A]]$.

From corollary 8.4.5, $\overline{S} \vdash \bigcirc A \Rightarrow S \xrightarrow{1} A$, where $A$ corresponds to the state notation of formula $A$ in the net. Consider $S \xrightarrow{1} A$. Indeed, the sequence of transitions from $S$ to $A$ can be expressed in a form $S \xrightarrow{0} S' \xrightarrow{1} S'' \xrightarrow{\delta} A$. Note that $S'$ could be either the same as $S$ or different from it. From $S'[\delta] \xrightarrow{0} A$, we have $S'[\delta] \in [[A]]$, which also means $S' \in [[\bigcirc A]]$. Moreover, as $\exists S' \in [[\bigcirc A]], S \xrightarrow{0} S'$, we have $S \in [[\bigcirc A]]$ from the definition of download closure.

**Case:** $[[A \otimes B]] = \{ S[\overline{S} \vdash A \otimes B] \}$  
We first prove that if $S \in [[A \otimes B]]$ then $\overline{S} \vdash A \otimes B$.

$[[A \otimes B]] = \{ S[S\exists S_A \in [[A]], S_B \in [[B]], S \xrightarrow{0} S_A + S_B] \}$, from the definition.

$\iff \{ S[S\exists S_A, S_B; \overline{S_A} \vdash A; \overline{S_B} \vdash B; S \xrightarrow{0} S_A + S_B] \}$, from induction steps above.

$\iff \{ S[S\exists S_A, S_B; \overline{S_A} \vdash A; \overline{S_B} \vdash B; \overline{S} \vdash \overline{S_A} + \overline{S_B}] \}$, from corollary 8.4.5.

$\Rightarrow \{ S[S\exists S_A, S_B; \overline{S_A} \otimes \overline{S_B} \vdash A \otimes B; \overline{S} \vdash \overline{S_A} \otimes \overline{S_B}] \}$, applying $\otimes$ rule and the correspondence between $\otimes$ and $+$.

$\Rightarrow \{ S[S \overline{S} \vdash A \otimes B] \}$, applying cut rule.

Next we prove that if $\overline{S} \vdash A \otimes B$ then $S \in [[A \otimes B]]$.

$[[A \otimes B]] = \{ S[S \overline{S} \vdash A \otimes B] \}$

$\iff \{ S[S \xrightarrow{0} A \otimes B] \}$, from corollary 8.4.5.

$\Rightarrow \{ S[S \xrightarrow{0} A + B] \}$, from the correspondence between $\otimes$ and $+$.

$\Rightarrow \{ S[S \in [[A \otimes B]]] \}$, from the definition and given that $A \in [[A]]$ and $B \in [[B]]$. 

**Case:** $[[A \to B]] = \{ S | \overline{S} \vdash A \to B \} $

We first prove that if $S \in [[A \to B]]$ then $\overline{S} \vdash A \to B$.  

$[[A \to B]] = \{ S | \forall S_A \in [[A]], S + S_A \in [[B]] \}$, from the definition.  

$\iff [[A \to B]] = \{ S | \forall S_A \in [[A]], S + S_A \overset{0}{\to} S_B \in [[B]] \}$, from the definition of downward closure.

$\iff [[A \to B]] = \{ S | \forall S_A \in [[A]], \overline{S} + \overline{S_A} \vdash \overline{S_B}, \overline{S_B} \in [[B]] \}$, from corollary 8.4.5.  

$\iff [[A \to B]] = \{ S | \forall S_A \in [[A]], \overline{S} \otimes \overline{S_A} \vdash \overline{S_B}; \overline{S_B} \vdash B \}$, from the induction steps above and the correspondence between $\otimes$ and $+$.  

$\Rightarrow [[A \to B]] = \{ S | \overline{S} \otimes A \vdash B \}$, from choosing $S_A = A$ and applying cut rule.  

$\iff [[A \to B]] = \{ S | \overline{S} \vdash A \to B \}$.  

Next we prove that if $\overline{S} \vdash A \to B$ then $S \in [[A \to B]]$.  

$[[A \to B]] = \{ S | \overline{S} \vdash A \to B \}$  

$\iff [[A \to B]] = \{ S | \overline{S} \otimes A \vdash B \}$  

$\iff [[A \to B]] = \{ S | \forall S_A \in [[A]], \overline{S} + \overline{S_A} \vdash \overline{S_B}; \overline{S_B} \vdash B \} \text{, consider further } S_A \in [[A]] \text{ and from the induction steps above.}$  

$\Rightarrow [[A \to B]] = \{ S | \forall S_A \in [[A]], \overline{S} + S \vdash B \}$, applying cut rule.  

$\Rightarrow [[A \to B]] = \{ S | \forall S_A \in [[A]], S_A + S \overset{0}{\to} B \}$, from the correspondence between $\otimes$ and $+$.  

$\iff [[A \to B]] = \{ S | \forall S_A \in [[A]], S_A + S \in [[B]] \}$, from the definition of downward closure.  

$\iff [[A \to B]] = \{ S | S \in [[A \to B]] \}$, from the definition.

**Case:** $[[A \& B]] = \{ S | \overline{S} \vdash A \& B \}$

We first prove that if $S \in [[A \& B]]$ then $\overline{S} \vdash A \& B$.  

$[[A \& B]] = \{ S | S \in [[A]] \cap [[B]] \}$, from the definition.  

$\iff [[A \& B]] = \{ S | S \in [[A]]; S \in [[B]] \}$  

$\iff [[A \& B]] = \{ S | \overline{S} \vdash A; \overline{S} \vdash B \}$, from the induction steps above.  

$\Rightarrow [[A \& B]] = \{ S | \overline{S} \vdash A \& B \}$, applying $\&$ rule.

Next we prove that if $\overline{S} \vdash A \& B$ then $S \in [[A \& B]]$.  

$[[A \& B]] = \{ S | \overline{S} \vdash A \& B \}$  

$\iff [[A \& B]] = \{ S | \overline{S} \vdash A \& B; A \& B \vdash A ; A \& B \vdash B \}$  

$\Rightarrow [[A \& B]] = \{ S | \overline{S} \vdash A ; \overline{S} \vdash B \}$, applying cut rule.
\[ [[A \land B]] = \{S \mid S \in [[A]]; S \in [[B]]\} \]

\[ \iff [[A \land B]] = \{S \mid S \in [[A]]; S \in [[B]]\} \]

**Theorem 8.4.9. (Completeness)**

\[ [[\Gamma]] \subseteq [[A]] \implies \Gamma \vdash A. \]

**Proof:**

We firstly establish that for any formulas \( \Gamma \) and \( A \),

\[ \text{if } [[\Gamma]] \subseteq [[A]] \text{ then } 0 \in [[\Gamma \to A]]. \]

Indeed, from the definition.

\[ S \in [[\Gamma \to A]] \iff \forall S_\Gamma \in [[\Gamma]], S + S_\Gamma \in [[A]]. \]

If \( [[\Gamma]] \subseteq [[A]] \), then \( \forall S_\Gamma \in [[\Gamma]], S + S_\Gamma \in [[A]]. \) Because \( 0 + S_\Gamma = S_\Gamma \), we have \( 0 + S_\Gamma \in [[A]]. \)

Hence, we have \( 0 \) satisfies the property

\[ \forall S_\Gamma \in [[\Gamma]], 0 + S_\Gamma \in [[A]], \] which means \( 0 \in [[\Gamma \to A]]. \)

Now consider \( [[\Gamma \to A]] \). From Lemma 8.4.8, we have \( [[\Gamma \to A]] = \{S \mid S \vdash \Gamma \to A\}. \)

Furthermore, given that \( 0 \in [[\Gamma \to A]] \), we have

\[ 0 \vdash \Gamma \to A. \]

As \( 0 \) is a state of an empty multiset, \( 0 \) corresponds to no formula, and so \( 0 \vdash \Gamma \to A \) means \( \vdash \Gamma \to A \iff \Gamma \vdash A. \)

Therefore, we can conclude that if \( [[\Gamma]] \subseteq [[A]] \) then \( \Gamma \vdash A. \)

Hence from the soundness and completeness theorems, with respect to the interpretation, we have that

\[ [[\Gamma]] \subseteq [[A]] \iff \Gamma \vdash A. \]

**8.4.4 Discussion**

We have demonstrated how a mapping between a form of (place) timed Petri nets and a fragment of TLL (which does not contain \( \oplus \)) is obtained, and that this mapping is sound and complete. From the mapping, we believe that it is possible to form an effective procedure for constructing an equivalent (place) timed Petri net from a protocol specification in TLL.

Execution of the protocols can then be based on execution of the equivalent TPNs.
However, to use a richer fragment of the logic, from the results obtained in [Engberg and Winskel, 1990] between linear logic and Petri nets, it is likely to require a restriction on the Petri nets to be atomic nets and so similarly on timed Petri nets. An atomic net has a property that every transition leads to a non-empty multiset of markings. Also, this restriction on Petri nets enables such interpretations of \(!A\) as \(A \& 1\). Such restrictions on Petri nets and possibly timed Petri nets make it quite limited for application of timed Petri nets as an execution framework for our particular TLL specifications described in Chapters 3 and 4. Hence, we believe it is better to use inference systems of TLL as a basis for executing TLL specifications of interaction protocols.

8.5 Summary

In this chapter, we provided detailed examples of typical interaction scenarios among agents to demonstrate how interactions can be constructed from such specifications. We also briefly discussed how flexibility of interaction is gained in these examples. Following the examples, an analysis of the TLL modeling framework and flexibility gained was provided. We also explored an alternative approach to protocol execution that makes use of timed Petri nets. Our findings suggest that using a proof search systems in TLL is likely to be more appropriate for our specification framework.
Chapter 9

Conclusion and Future Work

We have investigated a framework for modeling and specification of interaction among agents in a resource-conscious context. We have also discussed a suitable execution framework for such specifications based on proof search in TLL. Overall, the frameworks for specification and execution of agent interaction aim at providing flexibility in interaction. This chapter concludes our work, discusses related work and describes possible future research.

9.1 Conclusion

Our research questions, as stated in Chapter 1, were:

1. What is an appropriate framework for specifying flexible interactions that naturally deal with resources with respect to time?

2. What is an appropriate execution framework for turning such specifications into flexible interaction?

3. What is a framework that enables agents to reason about their choices and changes from the environment to take advantages of opportunities and deal with exceptions?

Flexibility in agent interaction has been considered with respect to two criteria as described in [Yolum and Singh, 2002b]. One is to preserve agents’ autonomy in their interactive actions and remove unnecessary constraints such as in traditional approaches to protocol specifications. The other is to enable agents to take advantage of opportunities and/or deal with exceptions that arise due to ongoing changes in dynamic multi-agent systems.
In order to answer the research questions, our work has used temporal linear logic (TLL), which is resource-conscious and deals with time, to model agent interaction. Interaction protocols are specified declaratively and executed by means of proof search.

We have demonstrated in Chapter 3 how TLL modeling can capture the dynamics of resource use, such as resource consumption, resource relocation, change of resource ownership, and resource transformation. Other concepts such as updating states, actions, goals and commitments can also be expressed naturally in this framework. In particular, the modeling of base commitments as negative formulas in TLL reflects the duality and causal relationship between what is provided in terms of resources and actions and what is required as base commitments. The modeling of conditional commitments makes use of a linear implication to naturally express the one to one causal relationship from the conditions to the commitment part. The notion of pre-commitment is considered as a potential form of conditional commitment and is expressed as having variables over the agents which may participate in the conditional commitment.

The modeling of various interaction concepts gives rise to a specification framework based on TLL as described in Section 4.1 and Section 4.2. We take the approach that interaction protocols are to be structured via the commitments of the participating agents. Agents then perform means-end reasoning over commitments in order to fulfil them. Our approach also allows agents to negotiate over commitments with respect to their current goals and base commitments, which reflects their reactions to on-going changes. Hence, agent protocols are specified in terms of goals and base commitments and pre-commitments, which reflect the services agents can provide to each other.

Chapter 3, Sections 4.1 and 4.2 have provided answers to the first research question, as discussed in Section 8.2. Flexibility in agent interaction that can be gained from the specification framework includes

- using a declarative approach to specifying what is to be achieved (goals and base commitments) rather than specifying a sequence of interactive actions for agents to carry out;
- having choices (among things and in time) expressed explicitly in specifications so that agents can reason about them;
- removing unnecessary order constraints by a connective of concurrency \(\otimes\);
- using commitments as an abstraction mechanism to structure protocols and allowing
means-end reasoning over them.

The second research question is addressed in Section 4.3 and Chapter 7. An execution model is introduced which describes how agents exchange messages containing requests and proposals whose contents are goals or base commitments and pre-commitments respectively. Specifications can then be executed based on proof search. Specifically, agents reason about how to find a proof of their goals or base commitments based on the available resources, actions, capabilities and pre-commitments. Pre-commitments of other agents can be provided as services and will form conditional commitments via negotiation among agents, which generate further interactions. In these concurrent interaction threads based around conditional commitments, resources and actions as well as goals or base commitments can be partially handled by the mechanisms described in Chapter 6. We have demonstrated that an implementation of this execution framework is possible via pseudo-code descriptions in the Appendix and discussions on implementation issues in Chapter 7.

Flexibility in agent interaction is hence further provided by

- allowing agents to negotiate over commitments, to choose which ones that satisfy their needs and help to cope with changes, rather than making them follow pre-determined commitments;
- providing smooth integration between handling changes from the environment and protocol execution;
- allowing agents to partially handle resources, actions and goals or base commitments. Given that the division is done by the agents, partial handling gives agents flexibility in utilizing resources and actions and fulfilling goals or base commitments.

Chapter 5 addresses the third research question by the modeling of the internal choices of agents and indeterminate possibilities which represent changes from the environment as well as various strategies for agents in dealing with them. Such strategies can be reasoned about and exercised by agents by using extra inference rules. A framework based on the choice calculus is provided to enable agents to reason about choices and changes and hence enable them to take advantage of opportunities and handle exceptions.

Chapter 8 discusses strengths and weaknesses of our work. We have demonstrated that TLL is a good candidate to capture various concepts used in agent interactions, such as resources, capabilities, and commitments. Our specification and execution frameworks are
capable of bringing flexibility to agent interaction as mentioned above. There are several weaknesses in our approach, of which a significant one is modeling time aspects using TLL.

9.2 Related Work

In this section, we discuss related work. Section 9.2.1 refers to related work that also uses temporal linear logic and linear logic to model interaction and reasoning. Section 9.2.2 then corresponds to related work that uses commitments to achieve flexibility.

9.2.1 Temporal Linear Logic Based Modeling of Agent Interaction

The work in [Küngas, 2003] describes a model of cooperative problem solving among agents using a partial deduction technique built on linear logic. Resources and capabilities in [Küngas, 2003] are modeled as linear logic propositional formulas and linear implications respectively. Capabilities are modeled in a form of a sequent:

\[ \vdash A \rightarrow B \]

where A and B are multiplicative conjunctions of literals in linear logic. Also, the states of agents (current state and goal state) are modeled as a multiplicative conjunction of resources formulas and do not contain capabilities.

Partial deduction principally comprises a backward chaining step and a forward chaining step, expressed in linear logic as below [Küngas and Matskin, 2005; Küngas and Matskin, 2006b]: A forward chaining step is defined as

\[
\frac{B \otimes C \vdash G}{A \otimes C \vdash G} R_f(L_i)
\]

A backward chaining step is defined as

\[
\frac{S \vdash A \otimes C}{S \vdash B \otimes C} R_b(L_i)
\]

where \( L_i \) is a label of a linear logic sequent of the form \( \vdash A \rightarrow_{L_i} B \), and A, B, C and G are multiplicative conjunctions. S and G are multiplicative conjunctions corresponding to the current state and goal state of agents respectively.

Though our framework has similar modeling for certain types of resources and capabilities, it significantly extends the modeling of agent systems.
Modeling capabilities as extra-logical axioms (as in [Küngas, 2003]) is applicable to capabilities with unlimited applications. In reality, however, there are capabilities that can be applied only a limited number of times and their modeling should reflect that. We therefore model capabilities as linear implications from pre-conditions to post-conditions. Those capabilities with unlimited applications are denoted by !. Also, capabilities are included in the agent’s state in our framework, which is different from [Küngas, 2003; Küngas and Matskin, 2005; Küngas, 2004a]. The inclusion of capabilities in the agent state is important because the use of capabilities with limited applications needs to be monitored. For example, the system state should reflect how many logins remain (applications of the login capability) after a series of failures.

Furthermore, as discussed in Section 3.2 we cater for a much wider range of concepts in agent interaction, include the modeling of infinite resources, status information, actions, choices, indeterminate possibilities and more importantly the modeling of base commitments, pre-commitments and conditional commitments. Given its foundation for agent interaction, using commitment-based modeling is a significant stepping stone toward modeling flexible agent interaction.

In addition, [Küngas, 2004b] extends the modeling of resources, capabilities and state information to include the time dimension by using temporal linear logic. A capability is denoted as $I \rightarrow O$ where in $I$ and $O$ are formulas in conjunctive normal form and the time operator $\bigcirc$ is only be allowed in $O$. This is the only use of time operators in [Küngas, 2004b] and therefore only describes a fixed allocation of resources in time. Our framework includes also the time operators $\Box$ and $\Diamond$ to account for modeling with uncertainty in time. $\Box$ and $\Diamond$, together with $\bigcirc$, form a much richer framework to describe (fixed and flexible) temporal constraints and ordering. We also allow $\bigcirc$ to be included in describing both the pre-conditions and post-conditions ($I$ and $O$) of capabilities so that non-monotonic relationships in time between them can be captured. For example, our modeling of capability allows the preconditions to include a commitment to pay in the next three days while the post conditions say the customer can have the sale item the next day.

Partial deduction techniques [Küngas, 2003; Küngas and Matskin, 2005; Küngas, 2004a] can be used for agent reasoning to figure out the missing capabilities when moving from the current state to the goal state. This can also be achieved in our framework. In particular, the backward chaining and forward chaining steps correspond to series of applications of sequent
calculus rules as follows. A forward chaining step corresponds to:

\[
A \vdash A \quad \frac{B \otimes C \vdash G}{B, C \vdash G} \quad \frac{A \rightarrow B, A, C \vdash G}{A \rightarrow B, A \otimes C \vdash G} \otimes L
\]

A backward chaining step corresponds to:

\[
\frac{A \vdash A \quad B \vdash B}{A, A \rightarrow B \vdash B} \quad \frac{C \vdash C}{A, C, A \rightarrow B \vdash B \otimes C} \otimes L
\]

\[
S \vdash A \otimes C \quad \frac{A \otimes C, A \rightarrow B \vdash B \otimes C}{A \rightarrow B, S \vdash B \otimes C} \quad \text{cut}
\]

where \(A \rightarrow B\) corresponds to the capability in use. Thus by using the sequent calculus rules, proof search can detect the capability \(A \rightarrow B\) required for the change in states.

The partial deduction approach is revised and extended with the inclusion of the temporal operator \(\bigcirc\) and \(!\) in [Küngas, 2004b]. These steps can be similarly translated back to the standard sequent calculus rules of temporal linear logic and hence can be applied in our framework.

In addition, the partial deduction technique supports the detection of subgoals and helps solve the problem partially [Küngas and Matskin, 2005]. Partial deduction can generate transformations from the current state towards the desired goal states and vice versa. However, it can only provide parts of the link between the current state and goal state. If the whole path is not found yet between the states then partial deduction may not provide any concrete solution because whether the partial path will be a part of the final solution is not yet known.

Our approach to the partial handling of goals can be seen as orthogonal to the partial deduction approach. We provide a logical technique (Chapter 6) to divide goals (resources) into subgoals (resource parts). Therefore, if a solution for a subgoal (or a utilization of a resource part) is found, then this partial solution can be applied (completeness), and the remainder of the goal (or resource) can be processed later. Partial handling has an important advantage over partial deduction with respect to flexibility in agent interaction. Indeed, the division of goals or base commitments and resources or actions can be performed dynamically by agents, subject to the current context, rather than being pre-specified by human designers with limited awareness of possible changes. In other words, depending on
the interaction context, agents can flexibly choose a part of their goals or base commitments to fulfill and choose a part of the resources or actions available. More importantly, as discussed in Section 6.5, these can occur in a distributed manner among concurrent interactions.

Hence, it can be seen that our framework supports both partial deduction from goals or resources formulas - via the standard sequent calculus rules - and partial handling of them, which when being combined together potentially increases the overall interaction flexibility.

Another advantage of our framework is the use of capabilities. In [Küngas, 2003; 2004a; Küngas and Matskin, 2005], capabilities required can be matched against those of other agents as a starting point for symbolic negotiation, cooperative problem solving and for coalition formation [Küngas and Matskin, 2006a;b]. In our framework, a similar but more general form of negotiation is used. Capabilities are modelled in our framework as pre-commitments, which require resources or actions in exchange for a capability. This method not only tracks the usage of capabilities but also makes the negotiation process more effective and more realistic.

To conclude, our framework, compared to that of [Küngas, 2003; 2004a; Küngas and Matskin, 2005; Küngas, 2004b; Küngas and Matskin, 2006a;b] has many similarities in modeling resources and capabilities, partial deduction reasoning, symbolic negotiation and cooperative problem solving. Our framework has additional features which include a variety of concepts relating to time; partial handling of resources and goals, which can be used in conjunction with partial deduction; and using a richer mechanism for negotiation over services based on capabilities.

9.2.2 Commitment-based Interactions

Another line of work [Yolum and Singh, 2002a;b; 2004; Chopra and Singh, 2004; Mallya et al., 2003; Wan and Singh, 2003; Flores and Kremer, 2004; Mallya and Singh, 2007] that has similar approaches to flexible interactions are those based on the notion of social commitment. Generally, the notion of social commitment is utilized to capture the meanings of states and actions of protocols. As a result, agents can reason about interactive actions and events in terms of their effects on commitments. Interactions are then essentially the creation, manipulation, exchange and resolution of commitments in a distributed manner. Agents act to fulfill their commitments but are not blindly committed to them and hence a certain level of flexibility is maintained. The use of commitments also provides greater flexibility than in following predefined sequences of interactive actions.
In the following paragraphs, we discuss some issues related to commitment-based work on protocols and how our framework has addressed them.

**Approaches based on Commitment Machines**

A commitment is a formula [Singh, 1999; Venkatraman and Singh, 1999; Yolum and Singh, 2002a] commonly written as \( C(x, y, p) \), where the debtor agent \( x \) becomes responsible (committed) to the creditor agent \( y \) for satisfying the proposition \( p \). Operations on commitments include creation, discharge (when \( p \) holds), cancellation, release (of \( x \) from the commitment), assignment (transferring the commitment from \( y \) to another creditor) and delegation (transferring the commitment from \( x \) to another debtor). These operations are defined in terms of states of commitments. As a result, protocol actions have two effects: one is as intended by the protocol designers and users, and the other is the effect on the commitments of the various agents involved. This duality, whilst intuitive, can ultimately complicate the understanding of the protocols. Also, the modeling of discharge operation is often indirect [Singh, 2007].

In our framework, we take an alternative approach where we define commitments as the dual of the resources, actions and capabilities (RACs) that are required to fulfill them. The interaction between commitments and the RACs that fulfill them can then be handled directly in the logic via sequent calculus. The advantage of our approach is that we not only include actions but also include resources, capabilities and system properties in account for the fulfillment of commitments. In this aspect, commitments have much richer and more realistic meanings. Sections 3.3.2 and 3.3.4 on modeling commitments provide details. Our work can further be improved by examination of the modeling of the changes of debtor and creditor of commitments.

In [Yolum and Singh, 2002a], a commitment machine is given to capture the set of legal states for the protocol, the meanings of actions and the final states (i.e. those in which the protocol can terminate). These meanings are defined in terms of commitments. The meaning of states is the set of active commitments. Agents can enter into protocols by accepting active commitments in a state. Due to the explicit meanings, a new state can be logically inferred from the old state and a given set of actions. Specifying protocols via commitment meanings rather than specifying detailed sequences of actions to be performed is a step forward to achieve flexibility. To this end, our framework has also achieved similar results. [Yolum and Singh, 2002a] further provides a translation of a specification in terms of commitment
machine into a specification based on Finite State Machines (FSMs), which is an advantage from both execution viewpoint (FSMs are readily executable) and design viewpoint (changes to FSMs can be made at the commitment machine level).

Several meta-level rules are in place to govern the operation of commitments. These rules include that a commitment ceases to exist when its proposition \( p \) becomes true; that a conditional commitment ceases to exist when its condition \( c \) becomes true and its base commitment becomes active; and that a conditional commitment ceases to exist when its base commitment is fulfilled. Although such meta-level rules enable the operations on commitments to have correct and consistent effects on commitments, they demand non-logical axioms to be in place, acting as global constraints over the operations. In our framework, however, such meta-level rules are already reflected in the modeling of commitments and hence their enforcement is done directly by the sequent calculus rules. It should be noted that our modeling of commitments as a negation of formulas of the RACs required is natural and simple. Specifically, that a base commitment ceases to exist when it is fulfilled is simply handled in the logic as the removal of the relevant negative formulas together with the corresponding positive formulas of the RACs required to fulfill them. Also, it is simple to enforce that the fulfillment of the conditions in a conditional commitment will automatically lead to the activation of the base commitment. More details are discussed later and in Sections 3.3.2 and 3.3.4.

[Yolum and Singh, 2002b; 2004] improve further the commitment machine formalism by defining commitments and their operations using the event calculus [Kowalski and Sergot, 1986], which is a formalism to reason about events, instead of classical logic. In particular, predicates of the event calculus are used as the building blocks for constructing commitment-based protocols. Commitments are then represented as properties and operations on commitments are defined via temporal statements (axioms) about the corresponding changes in values of these commitment properties over time. Reasoning rules are provided also in the form of non-logical axioms to realize the operational semantics of commitments which includes, for example, that a commitment is discharged when its proposition holds. It is demonstrated in [Yolum and Singh, 2004] that the specification and execution framework can handle various exceptions, accommodate different sequences and skip actions as necessary.

However, there are restrictions such as no concurrency among actions and that base commitments and conditional commitments are not broken. Our framework does not have these restrictions. Also, whilst unexpected changes can be integrated into protocol runs by the
introduction of particular actions, how these actions affect commitments needs to be coded in advance, and hence anticipated. Similarly to [Yolum and Singh, 2002a], these approaches require an extra layer of handling for correct manipulation of commitments and hence create unnecessary overheads. Also their framework does not allow agents to dynamically form protocols via reasoning.

In [Chopra and Singh, 2004], non-monotonic commitment machines are proposed, in which the underlining classical logic is replaced with Nonmonotonic Causal Logic [Giunchiglia et al., 2004]. Protocol specifications then further include protocol-independent theories on representing commitments and dealing with their operations, which corresponds to an extra layer of processing. In many ways, this approach has similar drawbacks to the use of meta-level rules over commitments in [Yolum and Singh, 2002a]. The advantages of the approach include a careful modeling of systems in terms of causality, a clear distinction between the concepts of fluents and actions, a rich modeling of actions, an ability to capture the persistence of commitment (inertial Fluents), and modeling of non-deterministic causality (may or may not cause). For example, [Chopra and Singh, 2004] provides a simpler and more natural way than [Yolum and Singh, 2002a] to express the idea that returning goods within 10 days causes the customer’s commitment to pay to be canceled. Though our framework has a natural way of capturing causality via linear implication, our modeling does not include causal concepts such as “every fact that obtains is caused and every fact that is caused obtains” as in ([Chopra and Singh, 2004]).

[Mallya et al., 2003] introduces a time perspective into representing and reasoning about commitments. A discrete and time branching temporal logic, which is an extension of CTL, is used as a representation logic. This approach represents a richer set of time concepts, especially time intervals and the operator Until, compared with ours. Extending our modeling of time is one of our directions for future work. This approach associates operations on commitments with commitment predicates to keep track of these operations. It then defines formally the semantics for the language that expresses commitment and its operations. The semantics lays the groundwork for establishing domain-independent properties of commitments, and provides a mechanism for expressing deadlines and evaluating commitment satisfaction with respect to these. This approach also allows temporal quantification to be dealt with independently from the associated propositions.

[Singh, 2007] developed Generalize Commitment Machines (GCMs) which extend commitment machines. GCMs are specified by states, actions, an action theory (capturing effects of actions on states) for transitions between the states and a set of “good” states. GCMs
have a significant advantage over commitment machines in that they can also be applied to non-terminating protocols. Operations on commitments are defined over the effects on commitments, similarly to other approaches. Protocol messages also have semantics in GCMs such that they can be directly operationalized. Messages in our framework are similar, but we have not considered the formalization of messages in this thesis.

The modeling of conditional commitments [Singh, 2007] allows them to be discharged either when the condition becomes true or when the base commitment becomes true. However, this is inapplicable to the case when one party fulfills the base commitment only if the other party satisfies the conditions. This kind of conditional commitments is extremely useful to capture exchange activities in agent interaction. The problem is that this approach permits the case of having the base commitment fulfilled without prior satisfaction of the conditions. Also, the fulfillment of the base commitment removes the conditional commitment and in effect, removes the conditions required of the other party. Such issues are avoided in our framework because of the way that the relationship from the conditions to the base commitment is maintained.

GCMs can be translated into deterministic Büchi Automata (BA) [Thomas, 1990]. One advantage of these machines over Finite State Machines is that they can express non-terminating computations. Specifically, an acceptance condition for a BA can be given as visiting “good” states infinitely often rather than terminating in a given final state. With some restrictions on GCMs, the translation to BA is shown to be sound and complete. Our framework also explores similar translation of protocols in TLL into timed Petri nets for execution purposes but further work is still required to accommodate the full MCA logic fragment.

Commitment machines and similar approaches [Yolum and Singh, 2002a;b; 2004; Chopra and Singh, 2004; Mallya et al., 2003; Singh, 2007] have several general issues.

They normally require prior specification of the meanings of all states in the protocol, particularly for the status of commitments. It should be noted that agents enter these states and progress towards resolving the corresponding commitments, but they have no ability to negotiation about these commitments. These predefined legal meanings act as prior constraints on the variations and development of the protocol and hence reduce the flexibility of interaction.

We overcome these limitations in our framework by not pre-defining legal meanings for protocols to follow. Instead, in our framework, some commitments act like goals and naturally reflect the roles of an agent in the interaction. Other commitments are generated as a
result of interaction amongst agents. These commitments are specified as pre-commitments, i.e. potential commitments. By reasoning about the relevance of pre-commitments (Sections 4.3.7 and 7.2.1) to current goals and pending commitments, subject to availability of RACs and possibly other factors, agents can dynamically select commitments from potential commitments. By negotiation with other agents, the selected pre-commitments can be turned into commitments. This selection and negotiation mechanism produces potentially greater variations for the protocol and the execution of protocols can be much more dynamic.

Moreover, when it comes to merging or extracting protocols, approaches based on commitment machines would normally require a significant level of human involvement to synthesize the set of legal meanings. When adding or removing commitments from protocols, it is likely that the task of adjusting the legal meanings would also involve significant human involvement. In this aspect, these approaches do not promote agent autonomy. In particular, agents should be able to dynamically manage protocols, as the on-going interaction may involve different objectives and causes. Agents should also be able to add and remove commitments in an active protocol as changes from the environment or other agents may make some commitments obsolete or better resolved by the agents alone.

To address these concerns, our framework promotes a dynamic formation of protocols as changes in the set of active commitments might arise for a number of reasons. The formation of protocols generally begins by agents introducing commitments. This is equivalent to entering a particular state in a commitment machine. Agents then enter further commitments as they attempt to fulfill existing goals and commitments. Also, the need to exchange resources, or to fulfill existing commitments may introduce more commitments. Unexpected changes from the environment or from other agents may lead to changes in the agents’ current set of resources, capabilities and goals and consequently, the agents may undertake further commitments or abandon existing ones.

Our framework supports changes in the set of active commitment by providing agents the ability to introduce new commitments if suitable pre-commitments are negotiated successfully, and also the ability to break existing commitments (technically possible as discussed in Sections 3.3.2 and 3.3.4).

In addition, commitment machine approaches typically lack an intermediate concept for an agent to reason about before entering a conditional commitment with other agents. In our framework this concept is pre-commitment. Our framework makes use of pre-commitments to allow agents to reason about which conditional commitment to form. In particular, this provides a means for agents to know if the other agent(s) also wants the conditional commit-
ment (and hence is likely to fulfill its conditions) before forming the conditional commitment. This is useful when the agent has multiple options for a conditional commitment and wants the other agents to pick the one that they prefer. Should the other agent disagree on the proposed pre-commitment, it can counter-propose another pre-commitment of its own. This helps increase agent autonomy over conditional commitments.

Other Commitment-based Approaches

[Fornara and Colombetti, 2003] extends the work in [Fornara and Colombetti, 2002] on defining an Agent Communication Language (ACL) in which the semantics of communicative acts is based on commitments. Specifically, the execution of communicative acts is regarded as actions which manipulate commitments. [Fornara and Colombetti, 2003] specifies interaction protocols as a set of rules regulating the performance of communicative acts of the ACL. Interaction diagrams are the main artifacts that specify which action can be performed by which agent at every protocol stage. Application-independent soundness criteria are also proposed for verification so that the design of protocols can ensure some desirable properties for successful interaction.

Compared to our approach, [Fornara and Colombetti, 2003] appears to have several restrictions. For example, the approach requires a large amount of work for protocol designers rather than promoting autonomy for agents. This work includes identifying every state; associating with every protocol state the set of all facts that hold, including the set of all commitment objects, temporal proposition objects, variables and domain specific objects; and creating transitions according to the (appropriate) execution of communicative acts and environmental events.

Commitments are modeled as structures whose fields include content, status (unset, cancelled, pending, active, etc) and conditions. Commitment contents are separated from their status, which requires a mechanism to be in place to update the status according to changes in the content. Even though the modeling of commitments is very detailed, tasks involved in handling commitments, such as creating them, changing status, and adding conditions, are carried out by non-logical means, unlike our approach.

[Mallya and Singh, 2004; 2005; 2007] aim to provide a formal framework for merging and refining existing protocols for a given business process. The mechanism is an algebra based on commitments [Mallya and Singh, 2007]. Comparisons between protocols are made on the basis of checking similarity between their runs (executed sequences of states) using state
similarity functions. These are based on criteria such as having the same creditor for every commitment or having the same roles in all commitments. Run subsumption is defined as including matches for every state while preserving temporal order between states. Merging protocols is then a matter of deriving a protocol consisting of runs that subsume some runs in every component protocol. The “choice” of two protocols is a protocol consisting of runs that subsume a run from either of the component protocols. The merge and choice operators are then shown to be idempotent, commutative and associative. Merge also distributes over choice. Together they form an algebra over protocols which allows protocols to be combined to give more options and functionalities, and hence flexibility.

Though our work does not provide such an elegant formal framework for the merging and refinement of protocols, we do support modular design of protocols as sets of pre-commitments. In principle, protocols can be expanded, contracted and merged, by a corresponding manipulation over the set of potential commitments (pre-commitments). How to support such a modular design of protocols is discussed in Section 4.2.3.

Our approach has certain advantages over [Mallya and Singh, 2007; 2004; 2005]. In particular, these approaches do not take into account other important factors in protocol manipulation such as changes of resources and capabilities (apart from changes in commitments and their status). Furthermore, examining all runs of all component protocols for similarity checking and computing subsumption run sets are very likely to be inefficient, which makes it less attractive in practice. Our approach does not rely on runs but on the higher-level and more intuitive concept of pre-commitment. The highly demanding task of determining which runs are suitable for the new protocol is replaced by finding the relevant pre-commitments to negotiate about for inclusion in the protocol. Checking pre-commitments for relevance is discussed in Section 4.3.7. This checking can in fact be reduced to checking the relevance of particular sub-formulas in pre-commitments and goals, which is likely to be significantly more efficient.

**General Remarks on Our Improvements**

The improvements that our work has made to existing work on commitment-based approaches to flexible interactions are largely due to our use of TLL to model commitments, and express resource concerns, choices, concurrency and time constraints.

Firstly, as discussed in Section 2.6, representing protocols in TLL makes it easier and more natural to express and handle resources. In approaches that use classical logics, representing
resources directly as formulas would normally require extra handling of the formulas when resources are used up. In our framework, the consumption of resources can be simply handled by an automatic removal of the corresponding formulas, as in $A \otimes (A \rightarrow B) \vdash B$.

Secondly, base commitments in the above approaches are typically modeled without a specification of what is required to fulfil them. Also, there is a need to design specific operators to handle the resolution of base commitments with the appropriate resources. In our approach, base commitments are modeled constructively as duals of the required RACs. Base commitments and what is required to fulfill them then have a natural symmetry. TLL, unlike other logics like classical logic and modal logic, provides a simple way to express this relationship. For example, $\Gamma \otimes [\Gamma]^\perp = \Gamma \otimes \neg \perp \vdash \perp$.

Thirdly, conditional commitments are used to ensure that the respective base commitments only come to existence when the conditions are fulfilled. However, the condition is modeled using classical logic, and hence does not provide a direct mapping between the condition and base commitments. In our framework, this mapping is expressed via the TLL connective $\neg \circ$.

Fourthly, in those approaches that are based on other logics, expressing concurrency of processes would typically require extra handling to make sure that these processes are not required to occur in a particular sequence. $\otimes$ naturally expresses concurrency by ensuring that two processes must have separate contexts, which is reflected in the following sequent calculus rule

$$\frac{\Gamma \vdash A, \Delta \quad \Gamma' \vdash B, \Delta'}{\Gamma, \Gamma' \vdash A \otimes B, \Delta, \Delta' \otimes R}$$

Fifthly, the above-mentioned approaches to flexible interaction using other logics usually lack a mechanism to express and reason about choices by agents. For example, interpreting the $\lor$ operator as choice does not distinguish between choices that agents can decide and possibilities that agents do not know and cannot decide. In our framework using TLL, (internal) choices can be expressed via the connectives $\&$ and $\square$ and reasoned about in a distributed and timely manner using sequent calculus rules, as discussed in Chapter 5.

Sixthly, the approaches that use other logics also have difficulties in reasoning about changes. In our framework, such desirable reasoning about changes is provided by using TLL to express (identifiable) changes in the form of indeterminate possibilities and using choice calculus rules to predict changes and act on predictions in a distributed and concurrent manner (as discussed in Chapter 5). The connectives in TLL that naturally express indeterminate possibilities are $\oplus$ and $\bigcirc$. **Related Work**
Finally, together, reasoning about choices and reasoning about indeterminate possibilities can produce fruitful results. In particular, several strategies are possible for agents. These include deciding choices in advance or at their associated times, taking a safe approach by preparing all the possibilities or taking a bold approach by predicting the outcomes of possibilities and following them. As agents apply these strategies, they explore the trade-off between RACs reservation and having choices (or taking risks). While agents decide on some choices or indeterminate possibilities and leave decisions on others open, those decided will eliminate reservation for RACs and hence possibly reduce the interaction sequences. The open choices remain as flexibility that the agents can explore.

9.3 Future Work

We have discussed some weaknesses and limitations of our work in Chapter 8. Our future work partly aims to overcome them and also explore other potential outcomes. Major directions of further work are described as below.

9.3.1 Work Related to Modeling

Our current modeling of breakable commitments makes an explicit use of choices on the expected outcome as a means for agents to alter the outcomes and hence break the commitments. However, this only covers intensional breaking of commitments. Unintentional breaking of commitments is a useful notion in agent interaction, especially among cooperative agents. Further work will attempt to capture the notion naturally, possibly based on the use of indeterminate possibilities.

The existing connectives in TLL like $\otimes$, $\&$, $\oplus$, $\neg$ can be used for describing the relationships among base commitments. It is a natural step forward to enrich our model by allowing a greater range of base commitments, such as allowing a base commitment inside another base commitment. For example, an embedded commitment might be $(A \otimes com^+)^\perp$, where $com^+$ is a base commitment. Manipulation of commitments can then be performed via transformations such as De Morgan’s laws and together with manipulation of the necessary resources and actions to fulfill them via standard proof search systems. Further investigation in this area is desirable.

Modeling the location and ownership of resources and actions using variables (and quantification of these variables over the domain of agents), such as $\text{resource}@\text{agent}X_{\text{agent}Y}$, makes it easy to express the dynamic changes in location and ownership. However, we have
not investigated deeply issues related to the handling of variables in various domains such as resources.

As discussed in Section 8.2.5, there is a need to extend TLL to capture intuitively the notion of “until” and to address the notion of having unlimited copies (of the formulas) at a finite time. Moreover, the new logic should allow the statement of properties such as “having both A and B at the next time point implies having A at the next time point and having B at the next time point”:

\[ \bigcirc (A \otimes B) \vdash \bigcirc A \otimes \bigcirc B \]

and “having both A and B anytime implies both having A anytime and having B anytime”:

\[ \Box (A \otimes B) \vdash \Box A \otimes \Box B \]

which is convenient in the resource context but is not supported by TLL.

Furthermore, it is also important to be able to specify a time point that is unknown, but must be the same for several things (for synchronisation). In TLL, to specify that \( A \), \( B \) and \( C \) are at the same time point which is unknown in a formula like \( \bigcirc A \rightarrow (\bigcirc B \otimes \bigcirc C) \) requires grouping them as \( \bigcirc [A \rightarrow (B \otimes C)] \), which has a different meaning because the time at which \( \rightarrow \) can be applied is now unknown.

A further direction of future work is to express time constraints as \( \bigcirc^x A \), where \( x \) is a variable that can be chosen or unknown, at the logic level. Apart from being able to express a specific time point by a certain value of \( x \), this representation of time can be used to model the notion of “anytime” as \( \forall x \; \bigcirc^x A \), which means \( x \) is universally quantified over the time domain, and model the notion of “sometime” as \( \exists x \; \bigcirc^x A \), which means \( x \) is existentially quantified over the time domain. It can be seen that a particular time, “anytime” and “sometime” can be modeled in a consistent manner using the form of \( \bigcirc^x A \) where \( x \) is a variable.

Moreover, such a representation of time can easily express a duration by giving the time range for the variable \( x \). Being concurrent at a specific time is also easily captured as in \( \bigcirc^x A \rightarrow (\bigcirc^x B \otimes \bigcirc^x C) \), where \( A \), \( B \) and \( C \) are at the same time (i.e. the next \( x^{th} \) time point), which may be known or unknown. In addition, this representation can be used to express relative relationships with reference to an uncertain time point. For example, to specify that \( B \) and \( C \) occur one time point later than \( A \) but the exact times of \( A \), \( B \) and \( C \) are not known, we can use \( \bigcirc^x A \rightarrow (\bigcirc^{x+1} B \otimes \bigcirc^{x+1} C) \) where \( x \) is unknown and \( x \geq 1 \). Hence, the expressiveness of the representation \( \bigcirc^x \) makes it attractive for future investigation.
A significant trend in IT industry is using service-oriented computing. Services are usually designed according to existing business rules in the domain. Hence, making interaction among agents more service-oriented and reflect business practice is desirable. Agents can then be naturally considered in terms of resources and capabilities. Services of agents that can be provided to other agents would naturally arise from the agents’ capabilities and are encoded with the logic of business rules. Protocols can then be designed in this a service-oriented manner as constraints on the services.

In our approach, pre-commitments are essentially what agents can provide to help others (post-conditions) at a cost (pre-conditions), and hence can be thought as services. These services are then proposed and negotiated among agents with respect to what they can provide and what is required. Moreover, the notion of pre-commitment helps to bridge the gap between agent capabilities and services provided to others. In particular, a discussion on how pre-commitments and hence conditional commitment can be designed based on agent capabilities and knowledge of business rules about fair exchanges is provided in Section 4.2.3. It can be seen that our approach is a step toward service-oriented interaction based on utilizing agents’ resources and capabilities.

Further work will investigate how specifications of service-oriented interaction can be done directly at the level of capabilities. In addition, in [Küngas, 2003; 2004b; Küngas and Matskin, 2005], it has been demonstrated how agents can explore capabilities of each other to identify potential for cooperation. This suggests some further work along the lines of dynamically converting specified capabilities into agent services.

9.3.2 Work Related to the Execution of Specifications

A natural and important step is to build an execution platform for specifications in TLL. Once such a platform is in place, it is then possible to investigate mechanisms to verify specifications of protocols and examine properties of protocol execution such as safety, liveness and conservation of resources.

Another aspect is the evolution of protocol specifications. In multi-agent systems, agents may enter into various interactions involving different protocols at the same time. Also, agents may acquire new capabilities or services which might result in changes to protocol execution. Synthesizing these protocols enables agents to deal with dependencies among interactions effectively, and to reason about their commitments and the uses of resources in a consistent and more efficient manner.
Future Work

The precise method of agent reasoning is another item of future work. Agent reasoning involving backward- and forward-chaining has been investigated for linear logic in [Harland and Winikoff, 2004]. These techniques are quite relevant to agent reasoning as they realize proactive and reactive behaviors of agents. The development of backward-chaining and forward-chaining techniques for TLL is an open problem.

In addition, timed Petri nets (TPNs) are well known tools for modeling concurrent, and time dependent systems but specifications on TPNs do not enjoy the benefits of being declarative, natural and modular like specifications in TLL. Therefore, combining specifications in TLL and execution in TPNs brings the benefits of both and provides links to visual representations and existing tools and techniques on TPNs. TPNs have been shown in Chapter 8 to be models of a fragment of intuitionistic TLL. It is then possible to map specifications in this fragment of TLL onto TPNs and vice versa. Our future work will explore such mappings to derive procedures that can turn a specification in TLL into TPNs and possibly vice-versa.

Lastly, specifications in our TLL framework require some detailed knowledge about the logic. Clearly, it would be beneficial to have a visual tool similar to those for UML diagrams which would allow non-expert users to specify protocols without having to learn the details of the formulas themselves.
Appendix A

Proof

A.1 Proof of $\hat{A} \vdash_{cc} \circ^a A \&_r \circ^b 1$

Theorem A.1.1. Let $\Gamma$ be a formula in the MCA fragment and $\hat{A}$ be its split up w.r.t. $A$ in $\Gamma$. We have

$$\hat{A} \vdash_{cc} \circ^a A \&_r \circ^b 1$$

where $\&_r$ is a representative choice of $A$ in $\Gamma$, $a$ indicates the time of the existence of $A$, and $b$ is an appropriate time depending on the outcomes of choices in $\hat{A}$.

Proof: we firstly establish that $\&_r$ is also a representative choice of $A$ in $\hat{A}$ and then prove the theorem by induction on the structure of $\hat{A}$.

By definition of $\hat{A}$, all the choices related to $A$ in $\Gamma$ are copied into $\hat{A}$ without switching the order of their two sides. Hence, the sequence of decisions on choices that retains $A$ in $\Gamma$ also retains $A$ in $\hat{A}$ and vice versa. In other words, $\&_r$ is also the representative choice of $A$ in $\hat{A}$.

A proof by induction on the structure of $\hat{A}$ is as follows.

Base step: The structure of $\hat{A}$ is $A$. The representative choice of $A$ in $\hat{A}$ is $A \&_r 1$, where $\text{Cond}L_r \vdash 1$, which is immediately fulfilled. This also means $\&_r \vdash L$. We then have $A \vdash_{cc} A \&_r 1$ because

$$\frac{[\&_r \vdash L]}{A \vdash_{cc} A \&_r 1} \&_r \frac{[\&_r \vdash L]}{A \vdash_{cc} A \&_r 1} \&_r$$

which forms a valid proof as the condition is satisfied.
Proof of $\hat{A} \vdash_{cc} \bigcirc^n A \&_{r} \bigcirc^b 1$

Proof of the reverse direction $A \&_{r} 1 \vdash_{cc} A$ is trivial:

$$
\begin{array}{c}
A \vdash_{cc} A \\
\hline
A \&_{r} 1 \vdash_{cc} A \& R
\end{array}
$$

**Induction step:** We assume that the hypothesis holds for $\hat{A}^n$ of $n$ structure levels:

$$\hat{A}^n \vdash_{cc} \bigcirc^a A \&_{r} n \bigcirc^b 1$$

where $\&_{r} n$ is the representative choice. Denote its determining conditions $CondL_{r} n$ as $\vdash conditionL$ and $CondR_{r} n$ as $\vdash conditionR$.

We then consider two cases where each of the conditions holds. Note that $conditionL$ and $conditionR$ are mutually exclusive.

When $\vdash conditionL$, which means $[\vdash \hat{\varphi}_{r} n \rightarrow L]$, we have

$$\hat{A}^n \vdash_{cc} \bigcirc^a A$$

from

$$
\begin{array}{c}
\hat{A}^n \vdash_{cc} \bigcirc^a A \bigcirc^b 1 \\
\hline
\hat{A}^n \vdash_{cc} \bigcirc^a A \&_{r} n \bigcirc^b 1 & R
\end{array}
$$

and

$$
\bigcirc^n A \vdash_{cc} \hat{A}^n \vdash_{cc} \bigcirc^n A \&_{r} n \bigcirc^b 1 & L
$$

When $\vdash conditionR$, which means $[\vdash \hat{\varphi}_{r} n \rightarrow R]$, we have

$$\hat{A}^n \vdash_{cc} \bigcirc^b 1$$

from

$$
\begin{array}{c}
\hat{A}^n \vdash_{cc} \bigcirc^b 1 \bigcirc^a A \bigcirc^b 1 \\
\hline
\hat{A}^n \vdash_{cc} \bigcirc^a A \&_{r} n \bigcirc^b 1 \vdash_{cc} \hat{A}^n
\end{array}
$$

and

$$
\bigcirc^b 1 \vdash_{cc} \hat{A}^n \vdash_{cc} \bigcirc^n A \&_{r} n \bigcirc^b 1 \vdash_{cc} \hat{A}^n \& L
$$

We need to prove in the case that $\hat{A}^{n+1}$ has $n+1$ structure levels:

$$\hat{A}^{n+1} \vdash_{cc} \bigcirc^a A \&_{r(n+1)} \bigcirc^b 1$$
Proof of $\mathcal{A} \vdash_{cc} \lnot_a A \land_r \lnot^b 1$

Note that possible structures of $\mathcal{A}^{n+1}$ of $(n+1)$ structure levels are:

$\mathcal{A}^n$, $\mathcal{A}^n \&_1 1$, and $\mathcal{A}^n \oplus_1 1$.

The determining conditions $\text{CondL}_{r(n+1)}$ of the representative choices of $A$ in these structures are (respectively):

$\vdash \mathcal{A}^n \lnot \lnot A \land \lnot A \lnot \lnot L$, and $\vdash \lnot A \land \lnot L$.

The determining conditions $\text{CondR}_{r(n+1)}$ are (respectively):

$\vdash \mathcal{A}^n \lnot \lnot A \land \lnot A \lnot \lnot L$, and $\vdash \lnot A \land \lnot L$.

We consider the two conditions for each case of the structure of $\mathcal{A}^{n+1}$.

Case: $\mathcal{A}^{n+1} = \mathcal{A}^n$.

In this case, $a'$, $b'$ are $a + 1$ and $b + 1$ respectively as $\mathcal{A}^n$ is one time point earlier.

When $\vdash \lnot \lnot A \land \lnot A$, which means $\vdash \lnot A \lnot \lnot L$. We have the proofs as follows.

\[
\begin{align*}
\mathcal{A}^n \triangleright_{cc} \lnot_a A &\vdash \lnot \lnot L \\
\mathcal{A}^n \triangleright_{cc} \lnot_a A &\vdash \lnot \lnot L \\
\mathcal{A}^n \triangleright_{cc} \lnot_a A &\vdash \lnot \lnot L
\end{align*}
\]

When $\vdash \lnot \lnot A \land \lnot A$, which means $\vdash \lnot A \lnot \lnot R$. We have the proofs as follows.
Case: $\hat{A}^{n+1} = \hat{A}^n \& \bot 1$

In this case, $a'$ is $a$ and $b'$ is $b$ or $0$.

The proof for the $\hat{A}^{n+1}$ case can be obtained from the proof for the hypothesis and includes the following steps.

When $\vdash (\hat{\xi}_{1} \rightarrow L)$, these steps are as follows.

\[
\begin{align*}
\hat{A}^n & \vdash_{cc} O^n A \vdash_{\text{condition}} \\
\hat{A}^n \vdash_{cc} O^n A [\text{CondL} \vdash_{\text{condition}}] \\
\hat{A}^n \vdash_{cc} O^n A [\text{CondL} \vdash_{\text{condition}} \& (\hat{\xi}_{1} \rightarrow L)] \& R \\
\hat{A}^n \vdash_{cc} O^n A \&_{r(n+1)} O^b 1 [\text{CondL} \vdash_{\text{condition}}] \& L \\
\hat{A}^n \&_{1} \vdash_{cc} O^n A \&_{r(n+1)} O^b 1 \\
\end{align*}
\]

or at the rule $\& R$, the agent follows the other possibility.

\[
\begin{align*}
\hat{A}^n & \vdash_{cc} O^b 1 \vdash_{\text{condition}} \\
\hat{A}^n \vdash_{cc} O^b 1 [\text{CondL} \vdash_{\text{condition}}] \\
\hat{A}^n \vdash_{cc} O^b 1 [\text{CondL} \vdash_{\text{condition}} \& (\hat{\xi}_{1} \rightarrow R)] \& R \\
\hat{A}^n \vdash_{cc} O^n A \&_{r(n+1)} O^b 1 [\text{CondL} \vdash_{\text{condition}}] \& L \\
\hat{A}^n \&_{1} \vdash_{cc} O^n A \&_{r(n+1)} O^b 1 \\
\end{align*}
\]
and

\[
\begin{align*}
\circ^n A \vdash_{cc} \bar{A}^n \models \lnot \text{condition} L & \quad & \circ^n A \vdash_{cc} \bar{A}^n \models \left( \frac{\delta^1}{\circ} \right) \land \text{condition} L \\
\circ^n A \vdash_{cc} \bar{A}^n \models \left( \frac{\delta^1}{\circ} \right) \land \text{condition} L \otimes \left( \frac{\delta^1}{\circ} \right) & \quad \& L \\
\circ^n A \& r_{n+1} \circ^b 1 \vdash_{cc} \bar{A}^n \models \left( \frac{\delta^1}{\circ} \right) & \quad \& R \\
\circ^n A \& r_{r(n+1)} \circ^b 1 \vdash_{cc} \bar{A}^n \& 1 &
\end{align*}
\]

or at the rule \&L, the agent follows the other possibility.

\[
\begin{align*}
\circ^b 1 \vdash_{cc} \bar{A}^n \models \lnot \text{condition} R & \quad & \circ^b 1 \vdash_{cc} \bar{A}^n \models \left( \frac{\delta^1}{\circ} \right) \land \text{condition} R \\
\circ^b 1 \vdash_{cc} \bar{A}^n \models \left( \frac{\delta^1}{\circ} \right) \land \text{condition} R \oplus \left( \frac{\delta^1}{\circ} \right) & \quad \& L \\
\circ^n A \& r_{r(n+1)} \circ^b 1 \vdash_{cc} \bar{A}^n \models \left( \frac{\delta^1}{\circ} \right) & \quad \& R \\
\circ^n A \& r_{r(n+1)} \circ^b 1 \vdash_{cc} \bar{A}^n \& 1 &
\end{align*}
\]

When \( \vdash \left( \frac{\delta^1}{\circ} \right) \) does not hold, by mutual exclusion, this means \( \vdash \left( \frac{\delta^1}{\circ} \right) \) holds. The steps are (note that \( b' = 0 \))

\[
\begin{align*}
\text{CondR}_1 & \quad & [\text{CondR}_1, \vdash \left( \frac{\delta^1}{\circ} \right) R] \\
1 \vdash_{cc} 1 \left[ \text{CondR}_1, \vdash \text{condition} R \oplus \left( \frac{\delta^1}{\circ} \right) \right] & \quad \& R \\
1 \vdash_{cc} \circ^n A \& r_{r(n+1)} 1 \left[ \text{CondR}_1 \right] & \quad \& L \\
\bar{A}^n \& 1 \vdash_{cc} \circ^n A \& r_{r(n+1)} 1 &
\end{align*}
\]

\[
\begin{align*}
\vdash \left( \frac{\delta^1}{\circ} \right) R & \quad & [\vdash \left( \frac{\delta^1}{\circ} \right) R, \vdash \left( \frac{\delta^1}{\circ} \right) R] \\
1 \vdash_{cc} 1 \left[ \vdash \left( \frac{\delta^1}{\circ} \right) R, \vdash \text{condition} R \oplus \left( \frac{\delta^1}{\circ} \right) \right] & \quad \& L \\
\circ^n A \& r_{r(n+1)} 1 \vdash_{cc} 1 \left[ \vdash \left( \frac{\delta^1}{\circ} \right) R \right] & \quad \& R \\
\circ^n A \& r_{r(n+1)} 1 \vdash_{cc} \bar{A}^n \& 1 &
\end{align*}
\]

Hence, in all the cases of the conditions, and from the hypothesis we have

\[
\bar{A}^n \& 1 \vdash_{cc} \circ^n A \& r_{r(n+1)} \circ^b 1
\]

**Case:** \( \bar{A}^{n+1} = \bar{A}^n \oplus_1 1 \)
\( a' \) is in this case \( a \) and \( b' \) is \( b \) or 0.

Similarly, the proof for the \( \overline{A}^{n+1} \) case can be obtained from the proof for the hypothesis and includes the following steps.

When the outcome of \( \oplus_1 \) is the left hand side, which means \( \vdash (\overline{\alpha_1} \rightarrow L) \), we have the following proof steps.

\[
\frac{\overline{A}^n \vdash \alpha \vdash \text{condition}L}{\overline{A}^n \vdash \alpha \vdash L, \vdash \text{condition}L} \]
\[
\frac{\overline{A}^n \vdash \alpha \vdash L, \vdash \text{condition}L \otimes (\overline{\beta_1} \rightarrow L)}{\overline{A}^{n+1} \vdash \alpha \& \alpha_{r(n+1)} \vdash b \vdash \text{condition}L \oplus (\overline{\beta_1} \rightarrow L)} \& R
\]
\[
\frac{\overline{A}^{n+1} \vdash \alpha \& \alpha_{r(n+1)} \vdash b \vdash L \oplus L}{\overline{A}^{n+1} \oplus 1 \vdash \alpha \& \alpha_{r(n+1)} \vdash b \vdash L} \]

or at the rule \( \oplus R \), the agent follows the other possibility.

\[
\frac{\overline{A}^n \vdash \beta \vdash \text{condition}R}{\overline{A}^n \vdash \beta \vdash \text{condition}R} \]
\[
\frac{\overline{A}^n \vdash \beta \vdash \text{condition}R \oplus (\overline{\beta_1} \rightarrow R)}{\overline{A}^n \vdash \beta \vdash \text{condition}R \oplus (\overline{\beta_1} \rightarrow R)} \& R
\]
\[
\frac{\overline{A}^n \vdash \beta \vdash \text{condition}R \oplus (\overline{\beta_1} \rightarrow R)}{\overline{A}^n \vdash \alpha \& \alpha_{r(n+1)} \vdash b \vdash L \oplus \text{condition}R} \& R
\]
\[
\frac{\overline{A}^{n+1} \vdash \alpha \& \alpha_{r(n+1)} \vdash b \vdash L \oplus \text{condition}R}{\overline{A}^{n+1} \oplus 1 \vdash \alpha \& \alpha_{r(n+1)} \vdash b \vdash L} \]

and

\[
\frac{\alpha \vdash \text{condition}L}{\alpha \vdash \text{condition}L} \]
\[
\frac{\alpha \vdash \text{condition}L \otimes (\overline{\beta_1} \rightarrow L)}{\alpha \vdash \text{condition}L \otimes (\overline{\beta_1} \rightarrow L)} \& L
\]
\[
\frac{\alpha \vdash \text{condition}L \otimes (\overline{\beta_1} \rightarrow L)}{\alpha \vdash \text{condition}L \oplus \text{condition}R} \& L
\]
\[
\frac{\alpha \vdash \text{condition}L \oplus \text{condition}R}{\alpha \vdash \text{condition}L \oplus \text{condition}R} \& L
\]

or at the rule \( \oplus L \), the agent follows the other possibility.

\[
\frac{\beta \vdash \text{condition}R}{\beta \vdash \text{condition}R} \]
\[
\frac{\beta \vdash \text{condition}R \oplus (\overline{\beta_1} \rightarrow R)}{\beta \vdash \text{condition}R \oplus (\overline{\beta_1} \rightarrow R)} \& L
\]
\[
\frac{\beta \vdash \text{condition}R \oplus (\overline{\beta_1} \rightarrow R)}{\beta \vdash \text{condition}R \oplus (\overline{\beta_1} \rightarrow R)} \& L
\]
\[
\frac{\beta \vdash \text{condition}R \oplus (\overline{\beta_1} \rightarrow R)}{\beta \vdash \text{condition}R \oplus (\overline{\beta_1} \rightarrow R)} \& L
\]

or at the rule \( \oplus R \), the agent follows the other possibility.
When the outcome of $\oplus_1$ is the right hand side, which means $\vdash (\rightarrow \circ R)$, we have the following proof steps, (note that $b' = 0$).

\[
\begin{align*}
\vdash \exists_1 \rightarrow \circ R_1 & \quad \vdash (\rightarrow \circ R) \\
1 \vdash \circ_1 \rightarrow \circ R_1, \vdash \circ_1 \rightarrow \circ R & \quad \vdash (\rightarrow \circ R) \\
\end{align*}
\]

Hence, in all the cases of the conditions, and from the hypothesis we have

\[
\exists^n \oplus_1 1 \vdash \circ_{n-cc} \circ \exists A \& \exists_{r(n+1)} \circ b' 1
\]

$\square$
Appendix B

Implementation Remarks

B.1 Data Structure

/***DATA TYPE***/
INTEGER AGENTID; /* data type of agents' IDs */
AGENTID AGENTS[]; /* a list of all agents' id */
ALL = AGENTS[0]; /* constant represents that all agents are considered */

AGENT {
/* RESOURCE represents a multiset of formulas of the resources and actions that the agent has. */

. FORMULA RESOURCE;
    /* RULES[] represents a set of formulas of the pre-commitments that the agent has. */

. FORMULA RULES[];

. /* COMS[] represents a set of base commitments or goals of the agent which have not been fulfilled yet. */

. FORMULA COMS[];

. /* the array DECISIONS contains elements as tuples that each contains three elements - a commitment formula, a boolean expression that indicates if the commitment is fulfilled or */
not, and a boolean expression that tells if such indication is valid. Validity of the indication
depends on whether all the choices in the indication boolean expression have all been decided
or not. */

. (FORMULA, BOOLEAN, BOOLEAN) DECISIONS[];
}

/* end of AGENT definition */

/* we assume that choice are properly IDed and their identifications managed properly
so that each appearance of choice (\& or \oplus) in formulas is externally attached with its ID and
as a result, the choice’s other attributes can be retrieved appropriately. */

R = 1; /* constant representing a decision to choose right */
L = 2; /* constant representing a decision to choose left */
U = 3; /* constant representing an undecided choice */

CHOICE {
. INTEGER VALUE=L, R, U;
. FORMULA COND;
. INTEGER ID;
. INTEGER TIME; /* time of the choice */
. BOOLEAN decided;
}

EXPRESSION {
. BOOLEAN c_1;
. BOOLEAN c_2;
. VALUE={c_1 \rightarrow c_2}
}

FORMULA {
Logic formula which contains literals and connectives and operators \&\&, \oplus, \&, \bigcirc as in the
fragment MCA discussed in section 6.1. }

PROPOSITION {
Linear logic proposition which contains only an atom. }
B.2 Functions for Agent Interaction

B.2.1 How to Fulfill (Partly) a Goal/Base Commitment

```plaintext
/********************-FULFILL-********************/
/* FULFILL receives a resource formula (res) and a commitment formula (com⊥). FULFILL finds common basic linear logic formulas at the same time point among the two formulas. Based on such basic linear logic formulas, FULFILL attempts to resolve all the possible parts of the commitment using the resource formula provided, based on agents’ decision making on choices involved in both the resources and the commitment. FULFILL calls a function APPLY_STRATEGY to further resolve the sub-commitment using the matched resource to reflect the agent’s strategies in dealing with the choices involved. FULFILL returns TRUE if a matching is found between a part of the commitment formula and a part of the resource formula and FALSE otherwise.*/
BOOLEAN FULFILL(FORMULA res, FORMULA com⊥) {
    INTEGER idx_a, idx_b;
    PROPOSITION pro;
    FORMULA part_res; /* part of the resource formula that contains a basic linear logic formula and/or 1s */
    FORMULA part_com; /* similarly is the part for the commitment formula */
    BOOLEAN return;

    return := FALSE;
    pro := MATCH(res, com, idx_a, idx_b);
    WHILE (pro ≠ NIL) DO {
        SPLIT(res, part_res, idx_a);
        SPLIT(com, part_com, idx_b);
        APPLY_STRATEGY(part_res, part_com⊥);
        pro := MATCH(res, com, idx_a, idx_b); /* finding further common proposition */
        return := TRUE; /* the formula com⊥ may be fulfilled in part or wholly*/
    }
    RETURN return;
}
```
B.2.2 How To Make a Request

/******************** REQUEST_ ********************/

/* REQUEST receives an agent and a commitment formula com⊥. REQUEST makes a request for the commitment to the agentTo or to each agent in the operating environment if agentTo = ALL until the request is resolved or there is no more agent to send to. REQUEST returns TRUE if the commitment is resolved via the request and FALSE otherwise.

Replies to a request might be an indication of request failure or a proposal. In the later case, the success of the request depends on the success of the proposal. In case agentTo is ALL, having a failure indication for the request message or failure in the reply proposal, the agent can send the request to other agents. */

BOOLEAN REQUEST(AGENTID agentTo, FORMULA com⊥) {
  AGENTID requested_agent;
  BOOLEAN return;
  STRING reply;
  INTEGER agent_idx;
  BOOLEAN continue;
  BOOLEAN prop_succ; /* if a proposal is succesful */
  RULE pro_rule;

  return := FALSE;
  agent_idx := 1;
  continue := FALSE;

  IF (α ≠ ALL) THEN
    . requested_agent := agentTo;
  ELSE
    . requested_agent := AGENTS[agent_idx]; /* get the first agent in the list of all agents involved */

  /* consider two cases, one is for proposing to a single agent and the other is proposing to each agent until the proposal is carried out. */
  . REPEAT {
    . SEND(requested_agent, "REQUEST " + TO_STRING(com⊥));
Functions for Agent Interaction

. . . WAIT();
. . . reply := RECEIVE();

. . . /* if the reply indicate failure */
. . . IF (reply = "REQUEST " + TO_STRING(com⊥) + " FAILS") THEN {
. . . . . IF (agentTo = ALL) THEN {
. . . . . . . agent_idx := agent_idx + 1;
. . . . . . . requested_agent := AGENTS[agent_idx];
. . . . . . . IF (requested_agent ≠ NULL) THEN
. . . . . . . . continue := TRUE;
. . . . . . . ELSE
. . . . . . . . continue := FALSE;
. . . . . . }
. . . . ELSE /* reply is a proposal */
. . . . pro_rule := GET_RULE(proposal);
. . . . prop_succ := RECEIVE_PROPOSAL(owner, pro_rule, com⊥);
. . . . /* if the proposal is not successful */
. . . . IF NOT prop_succ THEN {
. . . . . IF (agentTo = ALL) THEN {
. . . . . . . agent_idx := agent_idx + 1;
. . . . . . . requested_agent := AGENTS[agent_idx];
. . . . . . . IF (requested_agent ≠ NULL) THEN
. . . . . . . . continue := TRUE;
. . . . . . . ELSE
. . . . . . . . continue := FALSE;
. . . . . . }
. . . . ELSE /* the proposal is fulfilled and hence the request is satisfied */
. . . . . return := TRUE;
. . . .}
. . .}ELSE /* the proposal is fulfilled and hence the request is satisfied */
. . . . return := TRUE;
. . }
. UNTIL (NOT continue);
. RETURN return;
### B.2.3 How To Make a Proposal

/* PROPOSE receives an agent (agent_to), a rule and a commitment (com⊥) (possibly NULL). There are two cases. If the variable agent_to refers to a particular agent, then the function makes a proposal forming out of the given rule to the agent agent_to. If agent_to = ALL, then PROPOSE sends a proposal to each agent until the proposal is accepted and carried out completely or there is no more agent to send to.

PROPOSE returns TRUE if the proposal is accepted and carried out completely and FALSE otherwise.

There are two stages of completing a proposal. The first is an acceptance of the proposal, which is indicated by the "ACCEPT" response. The second is a receipt of the conditions of the proposal (LEFT(rule)) from the proposed agent. After the two stages are completed, the proposing agent will carry out the proposal.

Note that com⊥ refers to a commitment of the request whose reply is the proposal. Hence, if the proposal is not a reply to a request, com⊥ is NULL and agent_to can be ALL. */

```plaintext
BOOLEAN PROPOSE(AGENTID agent_to, RULE rule, FORMULA com⊥) {
  AGENTID proposed_agent; /* the proposed agent */
  BOOLEAN return; /* the proposed agent */
  STRING reply;
  INTEGER agent_idx;

  return := FALSE;
  agent_idx := 1;

  IF (agent_to = ALL) THEN
    proposed_agent := AGENTS[agent_idx]; /* AGENTS is a list of all agents involved */
  ELSE
    proposed_agent := agent_to;

  /* consider two cases, one is for proposing to a single agent and the other is proposing to each agent until the proposal is carried out. */
  REPEAT {
    SEND(proposed_agent, "PROPOSE " + TO_STRING(rule) +
```
Functions for Agent Interaction

\[ \text{TO\_STRING}(\text{com}^\perp) \]
\[ \text{IF NOT WAIT}(\text{proposed\_agent}, \text{rule}) \text{ THEN} \]
\[ \text{break}; \]
\[ \text{reply} := \text{RECEIVE}(); \]
\[ \text{IF} (\text{reply contains } "\text{REJECT}" ) \text{ THEN} \{ \]
\[ \text{IF} (\text{agent\_to} = \text{ALL}) \text{ THEN} \{ \]
\[ \text{agent\_idx} := \text{agent\_idx} + 1; \]
\[ \text{proposed\_agent} := \text{AGENTS}[\text{agent\_idx}]; \]
\[ \} \]
\[ \} \]ELSE IF (reply contains "ACCEPT") THEN{ \]
\[ \text{IF NOT WAIT}(\text{proposed\_agent}, \text{rule}) \text{ THEN} \]
\[ \text{break}; \]
\[ \text{reply} := \text{RECEIVE}(); \]
\[ \text{IF} (\text{reply} = \text{TO\_STRING}(\text{LEFT}(\text{rule}))) \text{ THEN} \{ \]
\[ \text{APPLY}(\text{LEFT}(\text{rule}), \text{rule}); \]
\[ \text{SEND}(\text{proposed\_agent}, \)
\[ \text{TO\_STRING}(\text{OWNED}(\text{proposed\_agent}, \text{RIGHT}(\text{rule})))}; \]
\[ \text{return} := \text{TRUE}; \]
\[ \} \]ELSE { \]
\[ \text{IF} (\text{agent\_to} = \text{ALL}) \text{ THEN} \{ \]
\[ \text{agent\_idx} := \text{agent\_idx} + 1; \]
\[ \text{proposed\_agent} := \text{AGENTS}[\text{agent\_idx}]; \]
\[ \} \]
\[ \} \]
\[ \} \]
\[ \text{UNTIL} (\text{proposed\_agent} \neq \text{NIL}) \text{ and } (\text{agent\_to} \neq \text{ALL}); \]
\[ \text{RETURN return}; \]
B.2.4 How to Respond to a Request

/* RECEIVE_REQUEST receives a requesting agent and a commitment as a content of a request. RECEIVE_REQUEST finds among its interaction rules a relevant rule to propose to the requesting agent. Relevance for a rule here is based on whether the outcome of the rule (RIGHT(rule)) is relevant to the commitment. If a rule is found and the proposal of that rule is successful, the function returns TRUE. RECEIVE_REQUEST sends a message of failure indication and returns FALSE if no rule is found or all the proposals of all relevant rules fail. */

BOOLEAN RECEIVE_REQUEST(AGENTID agent_req, FORMULA com⊥) {
    INTEGER rule_no;
    INTEGER idx_res, idx_com;
    STRING reply;
    RULE rule;
    FORMULA rightR;
    PROPOSITION pro;
    BOOLEAN failed; /* if the request fails */
    BOOLEAN pro_succ; /* if the proposal is successful */

    rule_no := 0;
    failed := TRUE;
    rule := NEXTRULE(owner, rule_no); /* get first rule of the owner agent*/
    /* search for rules that match the commitment */
    WHILE (rule ≠ NIL) DO {
        rightR := RIGHT(rule);
        IF CHECK_RELEVANCE(rightR, com⊥) THEN {
            pro_succ := PROPOSE(agent_req, rule, com⊥);
            IF NOT pro_succ THEN
                rule := NEXTRULE(owner, rule_no);
            ELSE {
                failed := FALSE;
                rule := NIL; /* propose no more rule */
            }
        }
    }
[^1]ELSE
. . . rule := NEXT_RULE(owner, rule_no);
. }
. IF failed THEN
. . SEND("REQUEST " + TO_STRING(com) + " FAILS");
. RETURN NOT failed;
}

B.2.5 How To Respond to a Proposal

/********************-RECEIVE_PROPOSAL-********************/

/* RECEIVE_PROPOSAL takes as arguments an agent as the proposing agent, a proposed rule, and a commitment com⊥ (possibly NULL) that might be of the request whose rely is the proposal.
If com⊥ is not NULL, which means that the proposal is a reply to a request of the commitment com⊥. RECEIVE_PROPOSAL then checks if the proposed rule is relevant to the commitment. Relevance for a rule here is based on whether the outcome of the rule (RIGHT(rule)) is relevant to the commitment.
If com⊥ is NULL, which means that the proposal is not related to any previous requests. RECEIVE_PROPOSAL then searches any pending commitment to which the proposed rule is relevant. If there is such a commitment, the proposal is accepted. Else, the proposal is rejected. After the proposal is accepted, the agent will attempt to fulfill the proposal’s conditions. If the conditions are fulfilled, the formulas of the conditions are sent to the proposing agent. Else, the agent sends an indication of failure and the proposal is abandoned.
RECEIVE_PROPOSAL returns TRUE if there is a commitment to which the proposed rule is relevant and the conditions of the proposal are fulfilled and FALSE otherwise. */

BOOLEAN RECEIVE_PROPOSAL(AGENTID proposing_agent , RULE proposed_rule, FORMULA com⊥ ) {
. INTEGER idx_com, idx_a, idx_b;
. FORMULA pend_com;
. BOOLEAN found, success;
. BOOLEAN return;

. idx_com := 0;
267

Functions for Agent Interaction

found := FALSE;
return := FALSE;

/* if there is no particular commitment, find a relevant pending one */
IF (com⊥ = NIL) THEN {
   pend_com := NEXT_COM(owner, idx_com); /* get first pending commitment */
   WHILE (pend_com ≠ NIL) AND NOT found DO {
      IF CHECK_RELEVANCE(OWNED(owner, RIGHT(proposed_rule)), pend_com)
      THEN {
         SEND(agent_pro, "ACCEPT");
         found := TRUE;
         success := TRANSACTION(RESOLVE(owner,(LEFT(proposed_rule))⊥));
         IF success THEN {
            SEND(proposing_agent, TO_STRING(LEFT(proposed_rule)));
            return := TRUE;
         } ELSE {
            idx_com := idx_com + 1;
            pend_com := NEXT_COM(owner, idx_com);
         }
      } ELSE {
         SEND(proposing_agent, "REJECT"); /* the proposal is not relevant */
      }
   }
   ELSE {
      IF CHECK_RELEVANCE(OWNED(owner, RIGHT(proposed_rule)), pend_com) THEN {
         SEND(proposing_agent, "ACCEPT");
         success := TRANSACTION(RESOLVE(owner,(LEFT(proposed_rule))⊥));
         IF success THEN {
            SEND(proposing_agent, TO_STRING(LEFT(proposed_rule)));
            return := TRUE;
         }
      }
   }
}
Functions for Agent Interaction

B.2.6 How to Resolve a Goal/Base Commitment

RESOLVE receives as arguments an agent agent_host and a commitment formula. RESOLVE represents attempts of the agent agent_host to resolve the commitment in several ways.

The first way is for the agent agent_host to use internal resources to resolve the commitment. In particular, possible parts of the commitment formula are matched with resource formulas to find common linear logic formulas at the same time point. Such matching and resolution are as is described in FULFILL.

After this attempt, if the commitment formula is still remained wholly or partly, the agent agent_host will interact with other agent(s). Interaction can be initiated by a request or a proposal or both sequentially if the first one fails. The preference over a request or a proposal resides at the agent_host. For simplicity, we describe one case where making a request is following making a proposal if the first one fails to resolve the commitment completely.

Given the commitment unfulfilled, agent_host then finds an interaction rule that is relevant to the commitment. Such relevant interaction rule is then proposed to each agent in the operating environment until the proposal is accepted and carried out successfully or there is no more agent.

If after the attempt(s) of making proposals, the commitment still remains partly (or wholly), agent_host will make a request for such commitment to each agent in the environment until the request is fulfilled or there is no more agent.

After all, if the commitment still remained partly (or wholly) then RESOLVE returns FALSE, otherwise, it returns TRUE. */

BOOLEAN RESOLVE(AGENTID agent_host, FORMULA com) {
  RULE rule;
  INTEGER rule_no; /* keep track of the number of current rule */

. FUNCTIONS for Agent Interaction

. FORMULA rightR, res;
. INTEGER idx_res, idx_com;
. PROPOSITION pro;
. BOOLEAN return;
. BOOLEAN continue;

. rule_no := 1;
. return := FALSE;
. not_search_all := TRUE;

. FULFILL(agent_host.resource, com⊥); /* fulfill the commitment locally */
. use rules to resolve the commitment */
. REPEAT
  . not_search_all := FALSE;
  . rule := NEXT_RULE(agent_host, rule_no);
  . /* search all the rules until one can resolve the commitment */
  . IF (rule ≠ NIL) DO {
    . . . rightR := RIGHT(rule);
    . . . IF CHECK_RELEVANCE(rightR, com⊥) THEN {
      . . . . IF (PROPOSE(ALL, rule, NULL)) THEN {
        . . . . . res := OWNED(agent_host, RIGHT(rule));
        . . . . . FULFILL(res, com⊥); /* rightR is already added to resource */
        . . . . } ELSE {
          . . . . . FORMULA com_r; /* commitment corresponds to LEFT(rule) */
          . . . . . com_r := [LEFT(rule)]⊥;
          . . . . . TRANSACTION(RESOLVE(agent_host, com_r));
          . . . . . IF (com_r = NULL) THEN /* com_r is completely resolved */ {
            . . . . . . APPLY(LEFT(rule), rule);
            . . . . . . res := OWNED(agent_host, RIGHT(rule))
            . . . . . . FULFILL(res, com⊥); /* rightR is already added to resource */
            . . . . . } ELSE {
              . . . . . . NOT{return} := FALSE;
              . . . . . . not_search_all := TRUE /* maybe partial fulfillment, hence try more*/
              . . . . . . }
          . . . . . }
        . . . . }
      . . . }
    . . . ELSE {
      . . . . NOT{return} := FALSE;
      . . . . continue := TRUE /* attempt to apply the rule internally */
      . . . . FORMULA com; /* global commitment */
      . . . . com := [LEFT(rule)]⊥;
      . . . . TRANSACTION(RESOLVE(agent_host, com));
      . . . . IF (com = NULL) THEN /* com is completely resolved */ {
        . . . . . APPLY(LEFT(rule), rule);
        . . . . . res := OWNED(agent_host, RIGHT(rule))
        . . . . . FULFILL(res, com⊥); /* rightR is already added to resource */
        . . . . . }
      . . . }
    . . . }
  . . }

Functions for Agent Interaction
Utility Functions

In this section, we will describe the supporting functions for the functions described in the previous section for agent interaction. Our purpose is to demonstrate that an implementation platform is possible given an implementation of relevant proof search techniques. Hence, an implementation of these functions is outside the scope of the thesis and so many functions are mentioned with only their descriptions (without content).

/********************-TRANSACTION-********************/
/* Transaction receives a function with associated arguments and executes that function. If the execution is successful then TRANSACTION returns TRUE. Successful execution means that there is no exception that occurs during the execution and the executed function returns TRUE. Otherwise, TRANSACTION returns FALSE and undos all the changes introduced by the function/procedure f. */
BOOLEAN TRANSACTION(FUNCTION f) {
  . EXCEPTION exception;

  . . . . . ADD(res, agent_host.resource);
  . . . . . not_search_all := TRUE /* maybe partial fulfillment, hence try more*/
  . . . . . }ELSE
  . . . . . rule := NEXT_RULE(agent_host, rule_no);
  . . . . . }ELSE
  . . . }ELSE
  . . . rule := NEXT_RULE(agent_host, rule_no);
  . . . }
  . UNTIL (com⊥ = NIL) OR NOT not_search_all
  . /* Failed to resolve completely via internal interaction rules, try to request */
  . IF (com⊥ ≠ NIL) THEN
  .  . return := REQUEST(ALL, com⊥);
  . ELSE
  .  . return := TRUE;
  . RETURN return;
}
TRY

EXEC(f);
CATCH(exception);
IF exception ≠ NIL THEN
ROLLBACK;
RETURN (exception = NIL);
}

********************-WAIT-********************

/* WAIT receives as arguments an agent agentTo and a message msg. WAIT waits for the response from the agentTo in a certain amount of time. If the time limit is over, WAIT will send a message to the agent agentTo with the content is the "msg" + FAILS. In this case, WAIT returns FALSE. If WAIT receives a response within the time limit then it returns TRUE.

Note that, WAIT does not consume the response message but does the checking only. Hence, subsequent call to RECEIVE() still gets the response message that WAIT gets.*/

BOOLEAN WAIT(AGENTID agentTo, STRING msg) {
}

********************-SEND-********************

/* Procedure SEND takes as arguments an agent name, and a message msg. SEND will send a message of the content msg from agent α (host agent) to agentTo. A primitive function SEND_MESSAGE is assumed present and handles sending messages from one agent to another. We assume that the sending always succeeds.*/

SEND (AGENTID agentTo, MESSAGE msg) {
...SEND_MESSAGE(agent α, agentTo, msg);
}

********************-RECEIVE-********************

/* Function RECEIVE inputs an agent name and returns a message if it receives successfully from that agent or returns NULL otherwise.

RECEIVE will wait for a maximum σ time to detect signal. It checks for signal at every predefined period. It makes use of the primitive function RECEIVE_SIGNAL that receives and transforms signals from an agent into a message. CHECK_SIGNAL listens and detects any relevant signal. It returns TRUE if a signal is detected and FALSE otherwise.*/

STRING RECEIVE(AGENTID agentFr) {
...time_expire := σ;
...clock := TIME();
...WHILE CHECK_SIGNAL(agentFr) = FALSE {
........IF (TIME() > clock + time_expire) THEN
...........RETURN NULL;
........ELSE
...........WAIT(1);
...}
...msg := RECEIVE_SIGNAL(agentFr);
...RETURN msg;
}

B.3.1 Functions on Rules

/************************-LEFT***************************/
/* The function LEFT receives a rule and returns all formulas on the left side of the linear implication \( \rightarrow \) of the rule. */
FORMULA LEFT (RULE rule) {
}

/************************-RIGHT***************************/
/* The function RIGHT receives a rule and returns all formulas on the right side of the linear implication \( \rightarrow \) of the rule. */
FORMULA RIGHT (RULE rule) {
}

/************************-GET_RULE***************************/
/* GET_RULE receives a proposal message and returns the proposed rule inside the message. */
RULE GET_RULE(STRING proposal) {
}

/************************-NEXT_RULE***************************/
/* NEXT_RULE receives an agent and a number referring to the current position in the list of rules of that agent and returns a rule corresponds to the next position in the list. Also, the current position is increased by 1. */
RULE NEXT_RULE(AGENTID α, INTEGER rule_no) {
  . RULE r ;
. r := α.RULES[rule_no];
. rule_no := rule_no + 1;
. return r;
}

B.3.2 Functions for Commitments

/**********************************_ADD_COM_******************************/
/* ADD_COM adds to the agent’s pool of commitments a new commitment which is recorded */
/* as a tuple in the global variable DECISIONS[]. The first element of a tuple refers to the */
/* commitment. The second and third elements are boolean expressions. The second element is */
/* evaluated to TRUE if the commitment is fulfilled and FALSE otherwise. The third element */
/* is used to indicate if the second element is a valid indication. */
ADD_COM(FORMULA com⊥, BOOLEAN exp, BOOLEAN decided) {
}
/**********************************_NEXT_COM_******************************/
/* NEXT_COM receives an agent α and a position number idx_com and returns the next */
/* pending commitment in the list of pending commitments COMS[] of the agent α, counting */
/* from the given position number idx_com.* /
FORMULA NEXT_COM(AGENTID α, INTEGER idx_com) {
}

B.3.3 Functions for Choices

/**********************************_GET_CHOICE_******************************/
/* GET_CHOICE receives a formula “for” and a position number ”idx” and returns a data */
/* structure CHOICE corresponding to the connective (⊕ or & ) at that position number idx in */
/* the formula “for”. */
CHOICE GET_CHOICE(FORMULA for, INTEGER idx) {
}
/**********************************_CHECK_CHOICE_******************************/
/* CHECK_CHOICE receives a choice ”c” (data structure CHOICE) and a choice decision */
/* ”dec” and returns TRUE if the decision on the choice ”c” is the same as the decision ”dec” */
/* and FALSE otherwise. CHECK_CHOICE works under the condition that the choice ”c” has
been decided. */

BOOLEAN CHECK_CHOICE(CHOICE c, INTEGER dec) {
}

/********************-NEW_CHOICE-********************/

/* NEW_CHOICEID generates and returns a newly created choice of data structure CHOICE with a unique ID. */

CHOICE NEW_CHOICE() {
}

/********************-BOOL_EVALUATION-********************/

/* BOOL_EVALUATION takes as argument a string and considers it as if it is a boolean expression and evaluates the string to a boolean value and returns that value. The argument string "exp" may as well contains calls to functions that return boolean value. */

BOOLEAN BOOL_EVALUATION(STRING exp) {
}

/********************-GET_STRATEGY-********************/

/* GET_STRATEGY receives condition formula that contains a series of decisions on choices and indeterminate possibilities. GET_STRATEGY will determine the overall strategy of the agent toward all the choices in the condition formula.

GET_STRATEGY returns TRUE if the agent is willing to accept the conditions and FALSE otherwise. */

BOOLEAN GET_STRATEGY(FORMULA cond) {
}

B.3.4 Functions for Formulas

/********************-LENGTH-********************/

/* LENGTH receives a formula and returns the length, measured in the number of characters, of that formula. */

INTEGER LENGTH(FORMULA for) {
}

/********************-GET_CHAR-********************/

/* GET_CHAR receives a formula and a position number and returns the character inside the formula at that position number. */

STRING GET_CHAR(FORMULA for, INTEGER idx) {
Utility Functions

\{/********************-OWNED-********************/\n/* OWNED receives an agent and a formula and returns a formula which is formed by a multiplicative conjunction of all the parts of the formula that belongs to agent \(\alpha\).

Note that the ownership of a basic TLL formula is denoted by the subscript bearing the name of the agent that owns the formula.

If the formula contains additive conjunctions/disjunctions of basic TLL formulas owned by different agents, OWNED will split all these additive conjunctions/disjunctions. These additive conjunctions/disjunctions are turned into additive conjunctions/disjunctions which are formed out of a basic TLL formula with \(\bigcirc\)1. */
\} FORMULA OWNED(AGENTID \(\alpha\), FORMULA for) {
\}  

\{/********************-GET_TIME-********************/\n/* GET_TIME receives a formula ”for” and a position number idx. The function returns the absolute time (time points) associated with the basic LL formula in the formula ”for” and located at the position number idx.

Note that the time operator \(\bigcirc\) either is associated with a basic LL formula or a compound. In the latter, there are \(\bigcirc\)(s) immediately preceding an opening round bracket(s). These \(\bigcirc\)(s) are also counted together with those immediately before the proposition. */
\} INTEGER GET_TIME(FORMULA for, INTEGER idx_pro) {

. STRING char;
. INTEGER idx;
. INTEGER time; /* the number of next time points */

. idx := idx_pro - 1;
. time := 0;
. char := GET_CHAR(for, idx);

. /* get all the \(\bigcirc\) associated with the proposition */
. WHILE (idx > 0) AND (char \(\neq\) ”(”) DO {
. . time := time + 1;
. . idx := idx - 1;
. . char := GET_CHAR(for, idx)
/* traverse backward starting from the first "(" before the proposition */
WHILE (idx > 0) DO {
  . idx := idx - 1;
  . char := GET_CHAR(for, idx);
  . . . /* get all $\circ$ immediately before "(" */
  . . IF (GET_CHAR(for, idx + 1) = "(") THEN {
  . . . WHILE (idx > 0) AND (char = "$\circ") DO {
  . . . . time := time + 1;
  . . . . idx := idx - 1;
  . . . . char := GET_CHAR(for, idx);
  . . . } }
  . . }
  . RETURN time;
}

/* NEXT_PRO receives a formula and a position number idx_pos in the formula and returns
an immediate basic LL formula at or after that position inside the formula. */
PROPOSITION NEXT_PRO(FORMULA for, INTEGER idx_pos) {

  STRING char;
  INTEGER idx;
  PROPOSITION pro;

  pro := NIL;
  idx := proposition;
  char := GET_CHAR(for, idx);

  . /* ignore all the characters that do not belong to the proposition */
  . WHILE (char in [""", "", "\bigoplus", "\bigotimes", "]\]) DO {
  . . idx := idx + 1;
  . . char := GET_CHAR(for, idx);


```plaintext
REPEAT
  pro := NIL + char;
  idx := idx + 1;
  char := GET_CHAR(for, idx);
UNTIL (char in ["","",","","])
RETURN pro;
```
CHECK_RELEVANCE returns TRUE if the agent is willing to act on the assumption of \( \widetilde{A}_{\text{res}} \vdash \widetilde{A}_{\text{com}} \) and FALSE otherwise. */

```
BOOLEAN CHECK_RELEVANCE(FORMULA res, FORMULA com⊥) {
  . FORMULA part_res; /* part of the resource formula that contains a basic linear logic formula and/or 1s */
  . FORMULA part_com; /* the part for the commitment formula that contains a basic LL formula and/or 1s */
  . INTEGER idx_res, idx_com;
  . FORMULA baseLL; /* the common basic linear logic formula */
  . FORMULA cond; /* conditions for the derivation */
  . BOOLEAN return;

  . return := FALSE;

  . REPEAT
    . . baseLL := MATCH(res, com, idx_res, idx_com);
    . . IF (baseLL ≠ NULL) THEN {
      . . . SPLIT(res, part_res, idx_res);
      . . . SPLIT(com, part_com, idx_com);
      . . . cond := PROVE(part_res, part_com⊥);
      . . . IF (cond ≠ NULL) AND GET_STRATEGY(cond) THEN
        . . . . return := TRUE;
        . . . ELSE
          . . . . baseLL := MATCH(res, com, idx_res, idx_com);
          . . . }
      . . }
  . UNTIL (res = NULL)
  . RETURN return;
}
/********************-APPLY-STRATEGY-*******************/
/* APPLY_STRATEGY receives a resource formula (res) and a commitment formula (com⊥).
If by using the resource formula to fulfill the commitment formula requires some certain decisions on choices, the commitment, together with those decisions are added to the variable DECISIONS[] via ADD_COM for future reconciliation with other choices’ decisions.

Note that making the decisions on choices to have the commitment fulfilled by the resource
Utility Functions

is the agent’s internal deliberation. How the agent considers to apply a particular choice strategy is not considered in the scope of the thesis. */

APPLY_STRATEGY(FORMULA res, FORMULA com{ } )

/********************-SPLIT-********************/

/* SPLIT receives two variables and a position number and performs the splitting on the variable res into two parts with respect to the basic LL formula located at the position number idx. The part that contains only the basic TLL formula and 1s is assigned to the variable "split". The remaining part is assigned to the variable "res".

SPLIT firstly locates the basic TLL formula in the formula from the given position number of the starting point of the basic LL formula. It then uses two other functions to generate the split ups.

The underlying assumption is that formulas are all well formed formulas in which for each connective, its two operands are only basic TLL formulas (of the form $\circ x A$) or well formed formulas. Formally, well formed formulas are constructed as following:

1. $F = A$
2. $F = \circ F$
3. $F = (F & F) | (F \oplus F) | (F & F)$

where A is basic linear logic formula (proposition). */

SPLIT(FORMULA res, FORMULA split, INTEGER idx) { }

. PROPOSITION pro; /* the proposition of concern, at the position idx */
. INTEGER i;
. STRING char;
. INTEGER start, end; /* starting and ending positions of the formula $\circ pro$ */

. pro := NEXT_PRO(res, idx);
. /* find the end position of the proposition */
. i := idx + 1;
. REPEAT
. . char := GET_CHAR(res, i);
. . i := i + 1;
. UNTIL NOT (char in [a-z]) OR (char = NULL)
. end := i - 1;

. /* get all the $\circ$ associated with the proposition pro */
. i := idx - 1;
. REPEAT
. . char := GET_CHAR(res, i);
. . i := i - 1;
. UNTIL (char ≠ ○) OR (char = NULL)
. start := i + 1;

. /* char can only be in ["", "", "", "", "] */
. IF (char = NULL) THEN /* i.e i = 1 and the formula is of the form ○ pro */ {
. . split := res;
. . res := 1;
. }

. /* generate formula res := res − pro */
. res := GENERATE_RES(res, start, end);

. /* generate formula split := pro */
. split := GENERATE_SPLIT(res, start, end);

} /*--------------------------GENERATE_RES--------------------------*/

/* GENERATE_RES receives a formula res, starting and ending position numbers then
returns a formula that is a split up of the formula res with respect to the basic LL formula
located between the start and end positions and does NOT contain that basic LL formula.
If the basic LL formula is A then the split up is of the form res − A.

The steps for generating the split up res − A are:
1/ traverse left or right to get the connective whose ○ A is one of its operands
2/ get the other operand of the connective.
3/ if the connective is one of & , ⊕ or none then replacing A by 1
else (connective is ⊗), remove both ○ A and the connective. */
FORMULA GENERATE_RES(FORMULA res, INTEGER start, INTEGER end){

. INTEGER i;
. STRING char;
. i := start - 1; /* traverse to the left */
. char := GET_CHAR(res, i);
. IF (char = ")") THEN {
. /* focus on those characters within the brackets "(...)" */
. i := i + 1;
. char := GET_CHAR(res, i);
. WHILE (char ≠ ")") DO {
. /* character in ["\(\)””, ”⊕”, ”&”, ”⊗”, a-z"] */
. . res := REPLACE(res, idx, end, 1); /* replace the proposition by 1 */
. . . } ELSE IF (char = ⊗) THEN {
. . . . res := REPLACE(res, start, end + 1, NULL); /* remove the whole proposition with its time and the connective */
. . . . break;
. . . } ELSE {
. . . . i := i + 1; /* ignore characters that are not connectives */
. . . . char := GET_CHAR(res, i);
. . . . }
. . }
. }ELSE /* char in ["\(\)”", ”&”, ”⊗"] */ {
. . IF (char ≠ "\(\)”") THEN {
. . . res := REPLACE(res, idx, end, 1); /* replace the proposition by 1 */
. . . break;
. . } ELSE IF (char = ⊗) THEN {
. . . . res := REPLACE(res, start - 1, end, NULL); /* remove the whole proposition with its time and the connective */
. . . . break;
. . . }
. . }
. RETURN res;
}
/********************-GENERATE_SPLIT-********************/
/* GENERATE_SPLIT receives a formula "split", starting and ending position numbers then returns a formula that is a split up of the formula split with respect to the basic LL formula located between the start and end positions and contains that basic LL formula. If the basic LL formula is A then the split up is of the form \( \hat{A} \).

The steps for generating the split up \( \hat{A} \) are:
1/ traverse left or right to get the connective whose \( \bigcirc^x A \) is one of its operands
2/ get the other operand of the connective
3/a if the connective is either of \&or\oplus or none then replace the other operand by 1
3/b else (connective is \otimes), remove both the other operand and the connective.
4/a get the immediate opening and closing brackets for the additive conjunction/disjunction and assign the whole group as B, adjusting the values of start and end accordingly. If there is no more formulas outside apart from B, terminate.
4/b remove the immediate opening and closing brackets for the multiplicative conjunction and assign the whole group as B, adjusting the values of start and end accordingly. If there is no more formulas outside apart from B, terminate.
5/ repeat the process from step 1 for B instead of \( \bigcirc^x A \). */

FORMULA GENERATE_SPLIT(split, start, end) {
  INTEGER i;
  STRING char;
  STRING con; /* the current connective */
  INTEGER brack_no; /* the number of opening (closing) brackets that haven’t been matched by closing (opening) brackets */

  WHILE (GET_CHAR(split, start-1) \neq NULL) DO {
    i := start - 1;
    char := GET_CHAR(split, i);
    IF (char = "(") THEN {
      /* traverse forward to the corresponding closing bracket */
      REPEAT
        i := i + 1;
      REPEAT
        char := GET_CHAR(split, i);
        IF (char in ["\oplus", ",", "\&", "\bigcirc", ",", a-z]) THEN
          con := char;
    }
Utility Functions

. . . . IF (char = "(") THEN
. . . . . . brack_no := brack_no + 1;
. . . . IF (char = ")") THEN
. . . . . . brack_no := brack_no - 1;
. . . . UNTIL (char = ")") AND (brack_no = 1)
. . . /* i is the position of the closing bracket corresponding to the first opening bracket */

. . . . IF (con in ["⊕", ";", ">", NULL]) THEN {
. . . . . . split := REPLACE(split, end+2, i-1, 1); /* replace the other operand by 1 */
. . . . . . start := start - 1; /* now include the opening bracket */
. . . . . . end := end + 3; /* now cover also the operand 1 and the closing bracket */
. . . . } ELSE IF (con = ⊗) THEN {
. . . . . . split := REPLACE(split, end+1, i, NULL);
. . . . . . /* remove the opening bracket */
. . . . . . split := REPLACE(split, start-1, start, NULL);
. . . . . }
. . . /* char in ["⊕", ";", ">", NULL] */
. . ELSE IF (char ≠ NULL) THEN {
. . . con := char;
. . . bracket_no := 0;
. . . i := start - 1;
. . . /* traverse backward to reach the beginning of the other operand */
. . . REPEAT
. . . . i := i - 1;
. . . . char := GET_CHAR(split, i);
. . . . IF (char = ")") THEN
. . . . . . brack_no := brack_no + 1;
. . . . IF (char = ")") THEN
. . . . . . brack_no := brack_no - 1;
. . . . UNTIL (char = ")") AND (brack_no = 0)
. . . /* i is now at the opening bracket right before the other operand */
. . . . IF (con in ["⊕", ";", ">"])) THEN {


Utility Functions

. . . . split := REPLACE(split, i+1, start-2, 1); /* replace the proposition by 1 */
. . . . start := i; /* now include also the operand and the opening bracket */
. . . . end := end - [(start-2) - (i + 1)]; /* updating end position after the replacement of
[(start-2) - (i + 1) + 1] characters by 1. */
. . . . end := end + 1; /* now cover the closing bracket */
. . . }
. . . ELSE IF (con = ⊗) THEN {
. . . . split := REPLACE(split, i, start-1, NULL);
. . . . /* remove the other operand and opening bracket */
. . . . split := REPLACE(split, end+1, end+2, NULL);
. . . . /* remove the closing bracket */
. . . . split := REPLACE(split, end+1, end+2, NULL);
. . . . }
. . . }
. RETURN split; }

/******************-REP_CHOICE-******************/

/* REP_CHOICE receives a formula "for" that contains only a basic TLL formula, possibly inter connected with multiple 1s via connectives ⊕ and/or &. The function returns a choice that is a representative choice of the basic TLL formula. REP_CHOICE goes through each choice and indeterminate possibility in the formula and determines the decision for them (LEFT or RIGHT) so that the basic TLL formula is kept remained. The sequence of decisions such that TLL formula remains in the end (all choices are decided) is turned into the condition for deciding LEFT of the representative choice. Condition for deciding RIGHT is also determined accordingly.*/

CHOICE REP_CHOICEFORMULA for) {

. INTEGER pro_idx; /* starting position of the proposition */
. INTEGER i;
. INTEGER length; /* the number of characters of the proposition */
. STRING char;
. STRING cond;/* determining conditions for the representative choice */
. INTEGER dec; /* a decision on a choice */
. CHOICE c_cur;
. CHOICE c_rep;
BOOLEAN decided; /* whether all the choices in the formula have been decided */

i := 1;
pro_idx := 1;
decided := TRUE;

/* get to the position of the proposition */
WHILE (i ≠ LENGTH(for)) DO {
  char := GET_CHAR(for, i);
  IF (char in ["(" , ")", "⊕", "&", "□", "1"] ) THEN 
    i := i + 1;
  ELSE 
    break;
  
  pro_idx := i;
/* get the length of the proposition */
WHILE (i ≠ LENGTH(for)) DO {
  char := GET_CHAR(for, i);
  IF NOT (char in ["(" , ")", "⊕", "&", "□", "1"] ) THEN 
    i := i + 1;
  ELSE 
    break;
  
  length := i - pro_idx + 1;

i := 1;
WHILE (i ≠ LENGTH(for)) DO {
  /* bypassing the proposition */
  IF (i = pro_idx) THEN 
    i := i + length;

  char := GET_CHAR(for, i);
  /* encounter a choice */
  IF (char in ["⊕", "&"] ) THEN {

 Utility Functions

. . . c_cur := GET_CHOICE(for, i);
. . . IF (i \mid pro_idx) THEN /* IF the proposition is on the right hand side of the choice */
.. . . dec := R;
. . . ELSE
. . . . dec := L;
. . . . IF (cond \neq NULL) THEN
. . . . . . cond := cond + "\land" + "CHECK_CHOICE(c_cur, dec)";
. . . . ELSE
. . . . . . cond := "CHECK_CHOICE(c_cur, dec)";
. . . . decided := decided AND c_cur.DECIDED; /* requiring all choices having been decided */
. . . }  
. . i := i + 1;
. }  

. c_rep := NEW_CHOICE();
. c_rep.DECIDED := decided;
. c_rep.COND := cond;
. IF (decided) THEN
.. . . c_rep.VALUE := BOOL_EVALUATION(cond);
. ELSE
.. . . c_rep.VALUE := L; /* a default value, only valid when DECIDED is TRUE */
. c_rep.TIME := GET_TIME(for, pro_idx);
. RETURN c_rep; }  
/********************-MATCH-********************/

/* MATCH receives two formulas, and their respective inner position numbers, and compares the two formulas to see if they contains a common basic linear logic formula of the same time point. */

MATCH searches for each basic linear logic formula in the first formula for_a a basic linear logic formula in the formula for_b that is the same and at the same time point. If the search does not find any matching basic linear logic formula, it returns NIL. Otherwise, the function returns the matching common basic linear logic and the associated number positions of them in their respective formulas. */
GER idx_b) {
  i := 0;
  WHILE (i < LENGTH(for_a) ) DO {
    for_ta := NEXT_PRO(for_a, i);
    i := i + LENGTH(for_ta);
    IF for_ta ≠ NIL THEN {
      j := 0;
      /* search for each proposition in for_b */
      WHILE (j < LENGTH(for_b) ) DO {
        for_tb = NEXT_PRO(for_b, j);
        IF for_tb ≠ NIL THEN {
          IF (for_tb = for_ta) THEN {
            time_1 := GET_TIME(for_a, i);
            time_2 := GET_TIME(for_b, j);
            IF (time_1 = time_2) THEN {
              idx_a := i;
              idx_b := j;
              RETURN for_ta;
            }
          }
        }
        j := j + LENGTH(for_b);
      }
      RETURN NIL;
    }
  }
  RETURN NIL;
}
/********************-END-**********************/

B.4 CD Player & Book Example

Description

The example is similar to that of [Küngas, 2003].
John has 10 dollars and a CD and wants to listen to music at the 8th next time point but his CD player is broken. John also intends to return books at the third next time point and can return books of others in the same visit. It takes a period of one time point to arrive at the library.

Peter intends to spend a day with his friends. He needs 20 dollars to cover the expenses (beer) but only has 15 dollars. Peter also has a book to be return before the fifth next time point and the transportation fee is 10 dollars. Peter has skills and can repair John’s CD player within a period of one time point.

Therefore, John and Peter can help each other to achieve goals. A protocol for their interaction is following. John returns the book for Peter in exchange for 10 dollars cost of transportation. Peter fixes John’s broken CD player and we assume that Peter charges 20 dollars for repairing the CD player.

B.4.1 Specifying Protocol

The interaction protocol between John and Peter is specified in our TLL framework as below.

**John**

John has 10 dollars, a broken CD player and a CD available for use at any time.

\[ 10 \Box $@J \otimes \Box \text{broken}\_\text{player}@J \otimes \Box CD@J \]

John has a commitment (goal) of listening to music at the 8th next time point.

\[ (\Box^8 \text{music}@J)^\perp \]

Rule 1 - John can visit the library at the third next time point and return a book for 10 dollars. It takes a duration of one time point for John to arrive at the library.

\[ \forall X, \Box^3 [\text{book}@X \otimes 10\Box X \rightarrow \Box \Box \text{book}@L_X \otimes 10 \Box \Box J] \]

Rule 2 - John can use a CD and a CD Player to play some music.

\[ \Box [CD@J \otimes CD\_\text{player}@J \rightarrow \text{music}@J] \]

**Peter**

Peter has 15 dollars, and a book available at any time.

\[ 15 \Box $@P \otimes \Box \text{book}@P \]

Peter has commitments (goals) of having the book returned to library before the fifth next time point, and enjoying some beer at the 9th next time point.

\[ [\bigoplus_{t\geq0} (\Box^t \text{book}@L_P)] \otimes (\Box^9 \text{beer}@P)^\perp, \]

where \( \bigoplus_{t\geq0} \bigoplus \bigoplus A = A \oplus \bigoplus A \oplus \ldots \bigoplus A \)

Rule 3 - Peter can repair a CD player for 20 dollars and make it ready at the following
time point.
\[\forall X, broken_{player}@X \otimes 20@$P_r \rightarrow \Box \Box CD_{player}@X \otimes 20@$P_r\]

Rule 4 - Peter can return the book to the library but it costs him 10 dollars for transportation fee and a travel time of one time point.
\[\forall book@P_r \otimes 10@$P_r \rightarrow \Box \Box book@L_p\]

Rule 5 - Peter can buy a case of beer for 20 dollars.
\[\forall 20@$P_r \rightarrow \Box beer@P_r\]

B.4.2 Reasoning Prior to Interaction

Peter and John both look for a way to derive their respective commitments but they failed due to a lack of resources.

At John
In order to fulfill his commitment of \(\Box^8 music@J_j\), John can apply Rule 2 as follows:

\[
\frac{\vdash \Box^8(CD@J_j \otimes CD_{player}@J_j)}{\Box[CD@J_j \otimes CD_{player}@J_j \rightarrow music@J_j] \vdash \Box^8 music@J_j}
\]

The application requires the conditions \(\Box^8(CD@J_j \otimes CD_{player}@J_j)\). Given that
\[
\frac{\vdash \Box^8 CD@J_j, \vdash \Box^8 CD_{player}@J_j}{\vdash \Box^8 CD@J_j \otimes \Box^8 CD_{player}@J_j \otimes R}
\]
\[
\frac{\vdash \Box^8 CD@J_j \otimes \Box^8 CD_{player}@J_j}{\vdash \Box^8 (CD@J_j \otimes CD_{player}@J_j) \otimes (\otimes) R}
\]

John needs to derive \(\Box^8 CD@J_j\) and \(\Box^8 CD_{player}@J_j\).

John already has \(\Box CD@J_j\) but cannot derive \(\Box^8 CD_{player}@J_j\) by himself.

At Peter
To satisfy the commitment of \(\Box^9 beer@P_r\), Peter can apply Rule 5 as follows

\[
\frac{\Box^k 20@$P_r}{\Box 20@$P_r \rightarrow \Box beer@P_r} \vdash \Box^k \Box beer@P_r
\]

and \(\Box^k \Box beer@P_r \vdash \Box^9 beer@P_r, 0 \leq k \leq 9\).

The application of Rule 5 requires \(\Box^k 20@$P_r, 0 \leq k \leq 9\) or a commitment of \(\Box^k 20@$P_r\), \(0 \leq k \leq 9\).

Fulfilling a commitment of \(\Box^k 20@$P_r\), \(0 \leq k \leq 9\), with \(k\) can be any value of choice
is equivalent to fulfilling a commitment of $\bigoplus_{k \geq 0}^9 (\bigotimes^{k20\$@P_p})$.  

If the commitment of $\bigoplus_{k \geq 0}^9 (\bigotimes^{k20\$@P_p})$ is fulfilled partly by $15 \square 10\$@P_p$, then later, Peter will not have the money $\bigotimes^3 10\$@P_p$ to have the book returned. At that point, Peter has to backtrack and choose not to fulfill the commitment immediately.

Also, to satisfy the commitment of $\bigoplus_{i \geq 0}^4 (\bigotimes^i book@L_p)$, Rule 4 can be applied as follows

$$\begin{align*}
\vdash \bigotimes^{i-1} (book@P_p \otimes 10\$@P_p) \\
\square \left[ \text{book@P_p} \otimes 10\$@P_p \rightarrow \bigotimes^i \square \text{book@L_p} \right] \vdash \bigotimes^i \square \text{book@L_p}
\end{align*}$$

where $\bigotimes^i \square \text{book@L_p} \vdash \bigoplus_{i \geq 0}^4 (\bigotimes^i \text{book@L_p})$, $1 \leq i \leq 4$

Peter analyzes the conditions:

$$\begin{align*}
\vdash \bigotimes^{i-1} \text{book@P_p} & \vdash \bigotimes^{i-1} 10\$@P_p \otimes R \\
\vdash \bigotimes^{i-1} \text{book@P_p} \otimes \bigotimes^i 10\$@P_p & \otimes R \\
\vdash \bigotimes^{i-1} \left( \text{book@P_p} \otimes 10\$@P_p \right) & \otimes (\otimes) R
\end{align*}$$

Because Peter has the book, $\square \text{book@P_p} \vdash \bigotimes^{i-1} \text{book@P_p}$, $1 \leq i \leq 4$, Peter can look for $\bigotimes^{i-1} 10\$@P_p$, $1 \leq i \leq 4$ or equivalently a commitment of $\bigoplus_{i \geq 1}^4 (\bigotimes^{i-1} 10\$@P_p)$ to completely satisfy the conditions.

Hence, the following commitments are possibly in place.

John: $(\bigotimes^8 \text{CD_player@J_p})$.

Peter: $(\bigoplus_{k \geq 0}^9 (\bigotimes^{k20\$@P_p})$ and

$\left[ \bigoplus_{i \geq 0}^4 (\bigotimes^i \text{book@L_p}) \right]$ or (if the rule 4 is used) $\left[ \bigoplus_{i \geq 1}^4 (\bigotimes^{i-1} 10\$@P_p) \right]$.

(2)

### B.4.3 Construction of Interaction

#### Reparing the Broken CD Player

There are two ways to start this section of interaction.

First is that John can make a request for $\bigotimes^8 \text{CD_player@J_p}$ to other agents (including Peter).

J to P: REQUEST $(\bigotimes^8 \text{CD_player@J_p})$

Peter looks for a rule to derive $\bigotimes^8 \text{CD_player@J_p}$ and finds Rule 3 applicable.

An instance of the rule can be proposed to John (with $X = J$):

P to J: PROPOSAL $\square \left[ \text{broken_player@J_p} \otimes 20\$@J_p \rightarrow \bigotimes \square \text{CD_player@J_p} \otimes 20 \square 10\$@P_p \right]$.

The second way is that (in order to gain some money to fulfill its commitments of $\left[ \bigoplus_{k \geq 0}^9 (\bigotimes^{k20\$@P_p}) \right]$ or possibly $\left[ \bigoplus_{i \geq 1}^4 (\bigotimes^{i-1} 10\$@P_p) \right]$) Peter advertises its capabilities...
to John. An instance with $X = J$ of Rule 3 is proposed to John:

$$P \to J: \text{PROPOSE} \Box [\text{broken}_{\text{player}}@J_j \otimes 20$@J_j \rightarrow \Box \text{CD}_{\text{player}}@J_j \otimes 20 \Box $@P_r]$$

Either way, John’s commitment of $(\Box^8 \text{CD}_{\text{player}}@J_j)^\top$ can be achieved by applying an instance of Peter’s proposal at the $m^{th}$ next time point ($0 \leq m \leq 7$), $m$ is at John’s choice) as follows.

Denote Peter’s proposal as $\Box [L_p \rightarrow R_p]$ then

$$\vdash \Box^m L_p \Box [L_p \rightarrow R_p] \vdash \Box^m R_p$$

From $\Box^m R_p = \Box^m (\Box \Box^m \text{CD}_{\text{player}}@J_j \otimes 20$@J_j), $\Box^{m+1} \Box \text{CD}_{\text{player}}@J_j$ can be extracted and $\Box^{m+1} \Box \text{CD}_{\text{player}}@J_j \vdash \Box^8 \text{CD}_{\text{player}}@J_j, 0 \leq m \leq 7.$ \hspace{1cm} (3)

Given that, John will accept the proposal.

J to P: ACCEPT

Given the analysis of the proposal as in (3), the conditions of the proposal are:

$\Box^m L_p = \Box^m (\text{broken}_{\text{player}}@J_j \otimes 20$@J_j).

As

$$\vdash \Box^m \text{broken}_{\text{player}}@J_j \vdash \Box^m 20$@J_j \otimes R \vdash \Box^m (\text{broken}_{\text{player}}@J_j \otimes 20$@J_j) \vdash \Box^m (\text{broken}_{\text{player}}@J_j \otimes 20$@J_j)

John has a broken CD player and $\Box \text{broken}_{\text{player}}@J_j \vdash \Box^m \text{broken}_{\text{player}}@J_j$. Then there only requires $\Box^m 20$@J_j or a commitment of $(\Box^m 20$@J_j)$^\top$, $0 \leq m \leq 7$.

As $m$ is at John’s choice, fulfilling the commitment is then equivalent to fulfilling the commitment of $[\bigoplus_{m \geq 0}(\Box^m 20$@J_j)]$^\top$. \hspace{1cm} (4)

If John has enough money, Peter then applies his proposal and derives $\Box^{m+1} \Box \text{CD}_{\text{player}}@J_j$ and $\Box^m 20 \Box $@P_r, $0 \leq m \leq 7$ or

$$\bigwedge_{m \geq 0}(\Box^m 20 \Box $@P_r). \hspace{1cm} (5)$$

Hence, from (3) and (1), John now fulfills his commitment of $(\Box^8 \text{music}@J_j)^\top$.

If John does not have enough money, John can disclose its capabilities to Peter to seek more money as in section “Returning Book to Library by John”.

**Returning Book to Library by John**

This section is applicable if the commitment of $[\bigoplus_{i \geq 0}(\Box^i \text{book}@L_r)]^\top$ is present. In other words, Rule 4 is not used, which means that Peter does not return the book by himself.

The section can start in two ways.
Either Peter makes a request for its commitment \( \bigoplus_{i \geq 0}(O^i book@L_p) \) to John.

P to J: REQUEST \( (\bigoplus_{i \geq 0}(O^i book@L_p)) \)

John finds Rule 1, \( \bigcirc^3[book@P_p \otimes 10\$@P_p \rightarrow \bigcirc \square book@L_p \otimes 10 \square \$@J_j] \), applicable and proposes it to Peter.

The second way is that John does not have enough money and hence proposes Rule 1 to gain some money.

From this point, the two ways proceed similarly.

An instance of John’s Rule 1 with \( X = P \) is proposed to Peter.

J to P: PROPOSE \( \bigcirc^3[book@P_p \otimes 10\$@P_p \rightarrow \bigcirc \square book@L_p \otimes 10 \square \$@J_j] \)

Peter has a commitment of \( (\bigoplus_{i \geq 0}(O^i book@L_p)) \) and this proposal rule can help it achieve this goal as follows.

Denote the proposal rule as \( \square [L1 \rightarrow R1] \) then

\[
\vdash \bigcirc^3 L1 \quad \square [L1 \rightarrow R1] \vdash \bigcirc^3 R1_{\text{app}}
\]

\( (6) \) where \( \bigcirc^3 R1 = \bigcirc^3(\bigcirc \square book@L_p \otimes 10 \square \$@J_j) \), \( \bigcirc^4 \square book@L_p \) can be extracted and \( \bigcirc^4 \square book@L_p \vdash \bigoplus_{i \geq 0}(O^i book@L_p) \).

Hence,

P to J: ACCEPT

The conditions are, according to \( (6) \), \( \bigcirc^3 L1 = \bigcirc^3(book@P_p \otimes 10\$@P_p) \). Peter analyzes the conditions:

\[
\begin{align*}
\vdash \bigcirc^3 book@P_p & \vdash \bigcirc^3 10\$@P_p \circ R \\
\vdash \bigcirc^3 book@P_p \otimes \bigcirc^3 10\$@P_p & \circ (\circ) R \\
\vdash \bigcirc^3 (book@P_p \otimes 10\$@P_p) & \bigoplus_{i \geq 0}(O^i book@L_p)
\end{align*}
\]

Given that Peter has a book to be returned, \( \square book@P_p \vdash \bigcirc^3 book@P_p \). There requires only \( \bigcirc^3 10\$@P_p \) or a commitment of \( (\bigcirc^3 10\$@P_p) \) \( (7) \).

If Peter has enough money, he will fulfill the conditions. John then applies the proposal and gains 10 dollars at the third next time point, \( \bigcirc^3 10 \square \$@J_j \). \( (8) \)

As a result of the rule application, Peter fulfills his commitment of \( (\bigoplus_{i \geq 0}(O^i book@L_p)) \).

If Peter does not have enough money, the proposal can not be applied and there are still pending commitments at the end of interaction. In this case, the interaction will fail.
B.4.4 Sequences of Interaction

Two threads of the interaction “Repairing Broken CD Player” and “Returning Book to Library by John” can occur in any order. This can be seen via a summarized version of the interaction as below. The question of whether these threads are successful (and so is the interaction), which means that agents have enough money to exchange for the services, is also answered.

Before the interaction

From (2),
At John, [state]: 10 □$@J \otimes □\text{broken\_player}@J_j$; [commitment]: $(\bigcirc^8\text{CD\_player}@J_j)^\bot$; [rules used]: 2.
At Peter, [state]: 15 □$@P_p \otimes □\text{book}@P_p$; [commitment]: $[\bigoplus_{k \geq 0}(\bigcirc^k20\$@P_p)]^\bot$ and $[\bigoplus_{i \geq 0}(\bigcirc^ibook@L_p)]^\bot$ or (if Rule 4 is used) $[\bigoplus_{i \geq 1}(\bigcirc^{i-1}108\$@P_p)]^\bot$; [rules used]: none.

We consider firstly the sequence “Repairing Broken CD Player”, then “Returning Book to Library by John”.

Repairing Broken CD Player

(Optional) J to P: REQUEST $(\bigcirc^8\text{CD\_player}@J_j)^\bot$
P to J: PROPOSE □[\text{broken\_player}@J_j \otimes 20\$@J_j \rightarrow □\text{CD\_player}@J_j \otimes 20\$@P_p]$
J to P: ACCEPT

At John, as John only has 10 □$@J_j \vdash \bigoplus_{m \geq 0}(\bigcirc^m10\$@J_j)$, in order to fulfill the proposal’s conditions, from (4), he must make a commitment of $[\bigoplus_{m \geq 0}(\bigcirc^m10\$@J_j)]^\bot$.
Hence, [state]: none; [commitment]: $[\bigoplus_{m \geq 0}(\bigcirc^m10\$@J_j)]^\bot$; [rules used]: 2.

At Peter, from (5) and given that $\bigwedge_{m \geq 0}(\bigcirc^m20\$@P_p) \vdash \bigoplus_{k \geq 0}(\bigcirc^k20\$@P_p)$,
[state]: 15 □$@P_p \otimes □\text{book}@P_p$; [commitment]: $[\bigoplus_{i \geq 0}(\bigcirc^ibook@L_p)]^\bot$ or (if Rule 4 is used) $[\bigoplus_{i \geq 1}(\bigcirc^{i-1}108\$@P_p)]^\bot$; [rules used]: 3.

Returning Book to Library by John

(Optional) P to J: REQUEST $[\bigoplus_{i \geq 0}(\bigcirc^ibook@L_p)]^\bot$
J to P: PROPOSE $\bigcirc^3[\text{book}@P_p \otimes 108\$@P_p \rightarrow □\text{book}@L_p \otimes 10\$@J_j]$
P to J: ACCEPT

At John, as from (8) and $\bigcirc^310\$@J_j \vdash \bigoplus_{m \geq 0}(\bigcirc^m10\$@J_j)$,
[state]: none; [commitment]: none; [rules used]: 2, 1.

At Peter: from (7) and given that $10\$@P_p \vdash \bigcirc^310\$@P_p$,
[state]: 5 □$@P_p$; [commitment]: none; [rules used]: 3.

The interaction ends successfully by both parties fulfilling all of their commitments.
We consider further the sequence “Returning Book to Library”, then “Repairing Broken CD Player”.

**Before the interaction**

At John, [state]: 10 $\square@J \otimes \square \text{broken}_\text{player}@J$; [commitment]: \((\bigcirc^8 CD_\text{player}@J)\perp\); [rules used]: 2.

At Peter: [state]: 15 $\square@P \otimes \square \text{book}@P$; [commitment]: \([\bigoplus_{k \geq 0}(\bigcirc^k \text{20@P})]\) and \([\bigoplus_{i \geq 0}(\bigcirc^i \text{book}@L_p)]\) or (if Rule 4 is used) \([\bigoplus_{i \geq 1}(\bigcirc^{i-1} 10@P_p)]\); [rules used]: none.

**Returning Book to Library by John**

(OPTIONAL) P to J: REQUEST \([\bigoplus_{i \geq 0}(\bigcirc^i \text{book}@L_p)]\)

J to P: PROPOSE \(\bigcirc^3 \text{book}@P_p \otimes 10@P_p \rightarrow \square \text{book}@L_p \otimes 10 \square \@J\)

P to J: ACCEPT

At John, from (8), [state]: 10 $\square@J \otimes \square \text{broken}_\text{player}@J \otimes \bigcirc^3 \text{10@J}$; [commitment]: \((\bigcirc^8 CD_\text{player}@J_j)\perp\); [rules used]: 2, 1.

At Peter: from (7), Peter has a commitment and it is resolved as 10 $\square@P \vdash \bigcirc^3 \text{10@P}$, [state]: 5 $\square@P$; [commitment]: \([\bigoplus_{k \geq 0}(\bigcirc^k \text{20@P})]\); [rules used]: none.

**Repairing Broken CD Player**

(OPTIONAL) J to P: REQUEST \((\bigcirc^8 CD_\text{player}@J_j)\)

P to J: PROPOSE \(\square \text{broken}_\text{player}@J \otimes 20@J \rightarrow \square CD_\text{player}@J \otimes 20 \square \@P\)

J to P: ACCEPT

At John, the commitment from (4) is resolved: 10 $\square@J \otimes \bigcirc^3 \text{10@J} \vdash \bigoplus_{m \geq 0}(\bigcirc^m \text{20@J})$, [state]: none; [commitment]: none; [rules used]: 2, 1.

At Peter: from (5) and given that \(\bigland_{m \geq 0}(\bigcirc^m \text{20@P}) \vdash \bigoplus_{k \geq 0}(\bigcirc^k \text{20@P})\), [state]: 5 $\square@P$; [commitment]: none; [rules used]: 3.

The interaction ends successfully by both parties fulfilling all of their commitments.

However, if Peter decides to return the book by himself, which means Rule 4 is applied. John will not return Peter’s book and hence can not gain 10 dollars from it. John therefore does not have enough money (20 dollars) for Peter to repair the broken CD player and consequently can not fulfill his commitment of \((\bigcirc^8 CD_\text{player}@J_j)\). Peter does not earn 20 dollars from repairing the CD player, and so has only 15 dollars. Hence, he can not fulfill all of his commitments of \([\bigoplus_{i \geq 0}(\bigcirc^i \text{10@P})]\) and \([\bigoplus_{k \geq 0}(\bigcirc^k \text{20@P})]\). Consequently, the interaction will fail.
B.4.5 Discussion

Partial deduction in [Küngas, 2003] allows John and Peter to associate all the unused resources with the missing resources or capabilities (money) for trading. For example, 10 dollars of John is deduced for getting the broken CD player fixed. However, a simple and direct implementation of the technique may not be fair to the proposing agent nor to the proposed agent in real life. In fact, if John has 300 dollars, the same partial deduction step would allocate the whole 300 dollars to the trading for the capability to repair the broken CD player.

In our approach, apart from allowing agents to figure out the missing capabilities, we predefine various desired offers that agent can make. This requires extra information, such as the price Peter would expect in return for fixing a broken CD player, to make sure that offers are reasonable.
Bibliography


U. Engberg and G. Winskel. Petri nets as models of linear logic. In A. Arnold, editor, CAAP ’90 - 15th Colloquium on Trees in Algebra and Programming Copenhagen, Denmark, May


