A Formal Semantics of Teamwork and Multi-agent Conversations as the Basis of a Language for Programming Teams of Autonomous Agents

Sanjeev Kumar
B.Tech., Electrical Engineering, Indian Institute of Technology, Kanpur (1995)
M.S., Computer Science and Engineering, OGI School of Science & Engineering at Oregon Health & Science University (2002)

A dissertation submitted to the faculty of the
OGI School of Science & Engineering at
Oregon Health & Science University
in partial fulfillment of the
requirements for the degree
Doctor of Philosophy
in
Computer Science and Engineering

June 2006

© Copyright 2006 by Sanjeev Kumar
All Rights Reserved

The dissertation "A Formal Semantics of Teamwork and Multi-agent Conversations as the Basis of a Language for Programming Teams of Autonomous Agents" by Sanjeev Kumar has been examined and approved by the following Examination Committee:

Philp R. Cohen
Professor
Oregon Health & Science University
Thesis Research Adviser

Milind Tambe
Associate Professor
University of Southern California

James Hook
Associate Professor
Portland State University

Mark P. Jones
Associate Professor
Portland State University

Peter Heeman
Assistant Professor
Oregon Health & Science University

Dedication

To my parents.

Acknowledgements

I would like to express my deep sense of gratitude towards my advisor, Phil Cohen, without whose unfailing help this thesis would not have existed. I feel grateful and privileged to have him as an invaluable mentor, colleague, and friend in this long journey. It has been a great learning experience, both as a researcher and as a person. Thank you Phil.

I would especially like to thank Milind Tambe for taking time to be the external examiner on my committee and for providing invaluable feedback through discussions, especially, during the early years of this research. I am extremely grateful to Mark Jones and Jim Hook for remaining on my committee even after moving to a different university, and for providing feedback from a programming languages perspective during the later part of my research. I would like to express my sincere gratitude towards Peter Heeman for agreeing to join my committee and read through this thesis at a very short notice.

There are many researchers with whom I have worked closely at various stages during my research and to whom I am indebted for shaping up my thinking. I would like to thank Hector Levesque for collaborating on group communication and extensions to joint intention theory. I would like to thank my friends and former colleagues Marcus Huber and David McGee for all the fun times we had brainstorming on different aspects of multiagent systems. I would like to thank Cynthia Breazeal and her team for sharing the internals of Leonardo, the robot, that provided the inspiration for the Lights World domain used in this thesis. I have benefitted immensely from discussions with Mike Wooldridge, Ray Perrault, David Israel, Tim Finin, James Allen, George Ferguson, Katia Sycara, Frank Dignum, Mark Greaves, Jeff Bradshaw, Yves Lesperance, Jeremy Pitt, Tim Norman, Karen Myers, David Pynadath, Amy Unruh, Paolo Busetta, Peter McBurney, Gal Kaminka and many other AI researchers who I have met at conferences and project meetings. Thank you all.

I have had the pleasure of being associated with the Center for Human Computer Communication (CHCC) during my years as a PhD student. CHCC provided a very stimulating research environment and I have had the privilege of working with some of the best researchers in the world. Thank you folks at CHCC and Natural Interaction Systems LLC, especially, Sharon Oviatt, Ed Kaiser, Matt Wesson, Ira Smith, Rachel Coulston, Courtney Darves, Paulo Barthelmess, Andrea Corradini, Rebecca Lunsford, Rajah Subramanian, Silvia Rossi and everybody else who has been part of the CHCC family at various times when I was there. You guys rock! And to my fellow students Ed, Rebecca, and Rajah, I would like to wish all the best for their forthcoming theses.

I am grateful to the Defense Advanced Research Projects Agency (DARPA) for financially supporting my thesis research through the CoABS Program (Contract No. F30602-98-2-0098, A0 G352), and through the Department of Interior, NBC, Acquisition Services (Contract No. NBCHD030010). I would also like to thank SRI for letting me work on parts of my thesis as part of the CALO project.

Finally, I am thankful to my parents for always being supportive and trusting my decisions. Their vision and sacrifice was the biggest inspiration that made it all possible.

Contents

List of Tables

List of Figures

Abstract

A Formal Semantics of Teamwork and Multi-agent Conversations as the Basis of a Language for Programming Teams of Autonomous Agents

Sanjeev Kumar

Supervising Professor: Philp R. Cohen

This dissertation demonstrates the feasibility of a logic-based declarative language for programming teams of autonomous agents that exhibit correct team and communicative behavior without having to program that behavior explicitly.

Teams tend to outperform any loose collection of individuals and are more robust to failures because team members coordinate as required and they communicate with each other appropriately for the success of the team as a whole. As such, the metaphor of teamwork is increasingly being employed to build intelligent systems consisting of distributed software entities (agents) that co-operate, coordinate, and communicate effectively as a team. However, teams of software agents are currently constructed by implementing predictions of teamwork theories in a very limited way due to the lack of a sound, comprehensive, and easily programmable approach for building such systems. Therefore, an important problem in multi-agent systems is the creation of a programming framework that enables teamwork and communication in a manner that bridges the gap between the theory and practice of these concepts.

This dissertation extends an existing formal theory of teamwork (Joint Intention Theory) by providing a comprehensive formal semantics of multi-agent communication based

on that theory along with support for a wider variety of teams. Thereafter, it presents a domain independent agent programming language called STAPLE with built-in support for teamwork and multi-agent conversations based on these theoretical contributions. STAPLE agents are programmed using a subset of modal logic, dynamic logic of actions, and temporal logic along with teamwork constructs and communication primitives that have a well-founded formal semantics.

The usefulness of STAPLE for programming teams of autonomous agents is demonstrated by showing that correct team and communicative behaviors follow from agent specifications in two different domains without having to program those behaviors explicitly in every possible situation. Firstly, the fault-tolerance specification of an agent architecture that is robust to sudden broker unavailability is provided to brokers written in STAPLE and the resulting STAPLE-based multi-agent system is shown to duplicate that fault-tolerant behavior. Secondly, STAPLE agents are shown to exhibit correct collaborative behavior in a simulated game that involves human-agent collaboration.

Chapter 1

Introduction

1.1 MOTIVATION

The growth of the Internet, the rapid explosion of connected mobile devices, the increasing acceptance of electronic commerce, and a software marketplace in which not one vendor develops all the required software components makes the problem of robust interaction, correct coordination, and well-understood, unambiguous communication between disparate software entities more important than ever before. Multi-agent systems are an important class of distributed systems consisting of multiple communicating software entities, or agents, that exhibit autonomous and goal directed behavior. One promising approach towards addressing the problem of robustness, coordination, and communication in multi-agent systems is to make the constituent agents work together as a close-knit team.

A team is more than a just a collection of individuals having a common, coordinated, and shared goal. For example, consider a group of motorists driving to the airport at any time. All of them have the common goal of reaching the airport in time for their respective flights. They share the goals of safely reaching their destination, they coordinate by properly following the traffic signals, and they may even help each other out, say, by making room for a motorist who wants to change lanes. However, one would hardly say that these motorists form a team. The reason is that even though the motorists have a common goal, they do not have a shared mental state. This difference is immediately obvious when one considers what happens when something goes wrong. For instance, if one of the motorists pulls over to the side of the road then we would not expect other

motorists to do the same. This behavior contrasts with that of a convoy which happens to be one of the most well known examples of teamwork [70]. Consider a convoy consisting of Bob and Harry. Assume that Bob knows the way to the airport and is leading the convoy but Harry does not know his way there and is following Bob. Convoys such as this are robust to various kinds of misunderstandings. For example, if Bob pulls over then Harry cannot assume that Bob no longer knows the way and continue on his own. Similarly, if Harry pulls over due to a problem in his car, then Bob cannot assume that Harry now knows his way and simply go away. Detailed discussions on the convoy example can be found in [70, 29].

We know from everyday experience that teams have several desirable properties – team members try their best to coordinate as required and to communicate with each other appropriately for the success of the team as a whole. As such, several formal theories of teamwork have been developed in the multi-agent systems literature [70, 91, 104, 45, 46] that constrain the behavior of individual teammates and address questions such as what to communicate, when to communicate, and with whom to communicate. One often used theory of team behavior in multi-agent systems is joint intention theory [70, 29], which prescribes a way to execute actions jointly by a team of agents. It requires that the team members be committed not only to their part of the joint action but also to the entire joint action. Therefore, they will not intentionally do something to render the performance of actions by other teammates impossible so as not to make the execution of the joint action impossible. In fact, each agent's commitment towards other agents' actions may lead to the agents helping each other in performance of their respective part of the joint action. Executing actions jointly in an uncertain environment, for instance a team of unmanned aerial vehicles jointly taking photograph of unfriendly territory in the presence of hostile fire and unsafe communication channels, is a situation where this behavior predicted by joint intention is very important. It follows from joint intention theory that two agents who are jointly committed towards bringing about a certain state of affairs will each have an individual commitment either to bring about that state of affairs, or to bring about mutual belief regarding the achievement, impossibility, or irrelevance of the jointly committed goal in accordance with the agent's private beliefs.

The specification provided by joint intention theory has been implemented in real world software applications demonstrating that teams as a whole waste less resources than a group of self-interested agents as the world gets more complex and unpredictable [52]. It has also been shown that a team tends to be more robust to failure than a collection of individual uncoordinated agents [109]. Furthermore, it has been shown that joint intention theory can be the basis of an agent communication language with a provably correct formal semantics [106, 107], and therefore it may provide a framework for communication and interoperation between disparate agents. These prior works on joint intention theory provide strong motivations for employing the teamwork metaphor towards addressing the problems of robustness, coordination, and communication in multi-agent systems.

However, the creation of teams of software agents currently requires the implementations of predictions of teamwork theories in a very limited way due to the lack of a sound, comprehensive, and easily programmable approach for building such systems. Therefore, an important problem in multi-agent systems is the creation of a programming framework that enables teamwork and communication in a manner that bridges the gap between the theory and practice of these concepts.

1.2 THE PROBLEM

The main problem addressed in this dissertation is the following: *Is it possible to achieve team and communicative behaviors in collaborative software systems without programming these behaviors explicitly, and if yes, how can we obtain these behaviors automatically*? In other words, we seek to investigate the feasibility of building a programming framework where one can declaratively specify joint action to get automatic team behavior along with correct task and team oriented communication. This framework should allow one to change the action specification declaratively to change the team behavior, and should enable one to predict team and communicative behavior offline and verify it by running the actual system.

1.3 HYPOTHESIS

We aim to provide a novel solution to the above problem by demonstrating the feasibility of a logic-based declarative language for programming teams of autonomous agents that exhibit correct team and communicative behavior without having to program that behavior explicitly.

The notion of an agent programming language is not new to the multi-agent systems literature [103]. Several agent programming languages [3, 36, 42, 44, 49, 50, 89] have been proposed in recent years that attempt to bridge the gap between logical theory of agency and agent implementations. However, most of these languages focus on individual agents and the support for agent teams and multi-agent communication is either non-existent or is at best ad-hoc. The early languages in this category did not have any support for communication, and they did not have agent modalities as independent declarative concepts. Formulas such as $\Diamond$p (eventually p) represented an agent's goal, and actions (or action expressions) represented an agent's commitment. The recent modifications of some of those languages do add a few declarative concepts like goal, as well as support for multi-agent communication. However, communication in these languages is usually an add-on feature not inherently related to the theory of agency behind the language. Moreover, none of these languages has built-in support for teamwork, at least not in the sense of formal connection to any teamwork theory.

On the other hand, joint intention (JI) theory [70, 29, 25] has successfully been used to implement several teamwork based applications and agent-development infrastructures [52, 63, 111]. It has also been used to provide formal semantics of communicative acts and multi-agent conversations [106, 107, 65]. However, the theory and the communication semantics have so far been used only for specification purposes. The rule-based team infrastructures implement the specifications of joint intention theory in a comprehensive and reusable manner demonstrating the usefulness of this theory, but a formal connection between the theory and its implementation is still missing.

We hypothesize that, by appropriately extending the logical theories of teamwork and communication, and by borrowing from research on planning and programming languages,

it should be possible to design and implement a domain independent language that provides teamwork and communication primitives such that the behavior of agents written in this language conforms to that predicted by the underlying theory. We present an agent programming language called STAPLE based on an extension of the joint intention theory and show that STAPLE supports teamwork and communication in a unified framework. Next, we outline the approach taken in this dissertation to investigate the above hypothesis.

1.4 APPROACH

We propose to follow a three phase approach in our investigation: theoretical, implementation, and validation. (1) First, we describe the basic teamwork theory (JI theory) and use it to specify semantics of speech acts and multi-agent conversations, showing how they can be used to create and discharge teams. This step includes enhancements to the JI theory itself to support groups of agents and a wider variety of teams. (2) Thereafter, we specify an agent programming language whose formal semantics derives from these theoretical contributions. We develop an interpreter for the single agent constructs of this language and then modify it to handle commitments of one agent towards another. (3) Finally, we argue that teams of agents written in this language coordinate and communicate as per the theory, without our having to program these behaviors explicitly. This claim is then verified by our using this interpreter to execute agents written in this language for two different domains.

The conceptual three phase process outlined above requires solving several problems that we discuss next.

1.4.1 Problems to be addressed

The prospect of supporting teamwork in a declarative programming language faces the following challenges.

1. *Reasoning about teamwork & communication:* The agents in a team created using existing methods of team creation respond using a fixed set of preprogrammed

messages, and hence are unable to act upon, reason with, and communicate about new messages and situations. The reason for this drawback is that there is no formal connection between the semantics of teamwork and that of the communicative acts. Therefore, agents cannot automatically start using a new communicative act simply by declaratively specifying the semantics of that communicative act. Furthermore, the agents in the earlier framework need to have their communicative behavior programmed explicitly because appropriate communication does not automatically follow from reasoning about teamwork.

2. *Interpreting logical constructs:* The constructs of teamwork theory and the communicative actions are defined in a logical language. However, there is no well known method of arriving at a programming language from the logical constructs and no established method of interpreting those constructs.

3. *Group communication:* The communication in a team inherently involves group communication. However, there is no formal semantics of group communication in the literature, especially one tied to the teamwork theory.

4. *Persistent teams*: The existing teamwork theories are concerned with one-time static teams that cease to exist once the team achieves it purpose. However, many teamwork applications require teams that persist but whose members may change dynamically with time.

5. *Multi-agent conversations:* Inter-agent communication is usually an extended conversation to achieve something rather then just a one-time communication that terminates with the exchange of a single message. However, there is no formal semantics for multi-agent conversations in the literature, especially within the context of teamwork, thereby making it impossible to prove the correctness of a conversation with respect to the current task.

6. *Testing & verification:* The teamwork theories provide specifications of agent behavior that can be hard-coded into an agent program. However, it is difficult to foresee all possible situations that agents will encounter and hence, it is difficult to predict a

complete behavior specification that works correctly in all possible agent interactions and situations. Implementing a new team specification in this old approach usually requires predicting the behaviors of agents using the teamwork theory and then implementing those predicted behaviors in the agent programs. As such, there is usually no quick way to test and verify a team behavior from its logical specification. We need implementations of useful teamwork and communicative behavior using the old approach to demonstrate that the same behaviors can be re-implemented easily and its variations can be tried out quickly in the new declarative language.

1.4.2 Proposed solution

This dissertation addresses the above problems by way of the following contributions.

1. *Reasoning about teamwork & communication:* It extends the prior work on agent communication languages based on the joint intention theory by redefining the formal semantics of communicative acts and showing that these communicative acts do result in creation and discharge of team commitments under appropriate conditions.

2. *Interpreting logical constructs:* It presents an agent programming language called STAPLE with built-in support for teamwork and multi-agent conversations by building upon a subset of the above theoretical contributions. It presents an implemented interpreter for STAPLE that directly executes agent specifications in a subset of modal logic, dynamic logic of actions, and temporal logic along with abstractions from the formal theory of teamwork and multi-agent conversations. The result is a domain-independent agent programming language formally connected with a logical theory of agency, whose constructs, including the communication primitives, have a well-founded formal semantics.

3. *Group communication:* It provides a formal semantics of group communication such that the individual communicative acts are a special case of the group communicative acts, and specifies the group communicative actions that are predefined in STAPLE.

4. *Persistent teams*: It extends the joint intention theory to include teams that continue

to exist even when the team membership changes. It provides a formal semantics for maintenance goals and a definition of teamwork based on maintenance goals that allows for teams that can continue to exist beyond one-time achievement goals.

5. *Acceptance test case:* It demonstrates the usefulness of persistent but dynamic teams by developing an agent architecture (called the Adaptive Agent Architecture or AAA) that implements a fault-tolerance specification based on this theory. It shows that multi-agent systems based on this architecture are robust with respect to certain kinds of broker failures without incurring undue teamwork overhead in the normal working conditions. The AAA provides a test case for the programming language proposed in this dissertation. AAA implements its fault-tolerant behavior using the old technique of team creation (by implementing the team specification). However, agents in the proposed language must easily replicate that fault-tolerant behavior upon providing them with its declarative specification.

6. *Testing & Verification:* It verifies that STAPLE does in fact satisfy the requirements mentioned earlier. It demonstrates the usefulness of STAPLE for programming teams of autonomous agents by showing that correct team and communicative behaviors follow from agent specifications in two different domains without explicitly programming those behaviors in every possible situation. First, the specification of a fault-tolerant agent architecture (AAA) that is robust to sudden broker unavailability is provided to brokers written in STAPLE. The resulting STAPLE-based multi-agent system is shown to duplicate that fault-tolerant behavior. Second, STAPLE agents are shown to exhibit correct collaborative behavior in a simulated game that involves human-agent collaboration.

7. *Multi-agent conversations:* It introduces a technique to specify formal semantics of conversation protocols within the framework of the joint intention theory by introducing a notion of landmarks. This technique allows treatment of conversation protocols as joint actions similar to that in natural language dialogue, and enables formal proofs about correctness of a conversation protocol with respect to its design goals. Finally, it demonstrates specification and direct execution of fully specified

conversation protocols in STAPLE programs.

Figure 1.1 illustrates the approach discussed above. In this figure, "subset" refers to the subset of aforementioned theoretical contributions that have been implemented in the STAPLE interpreter developed for this dissertation as follows:

1. *JI Theory extensions:* The implemented subset includes persistent but dynamic teams (i.e., teams whose membership can change and named teams). It excludes maintenance goals and teams jointly committed towards a maintenance goal.

2. *Semantics of communicative actions:* The implemented subset includes only those communicative actions that are actually used in the examples and test cases. It excludes other communicative actions such as "standing offer" whose semantics is presented in this dissertation.

3. *Semantics of group communication:* The implemented subset includes communicative actions required to establish and discharge joint commitments between persistent but dynamic teams whose membership is known at the time of performance of the communicative action. It excludes full-fledged group communication semantics in terms of "whoever" where senders and recipients are potentially unknown (as in a radio broadcast communication).

4. *Semantics of conversation protocols:* The implemented subset includes fully specified concrete conversation protocols (i.e., protocols in which all agents and all actions are known a priori). It excludes the treatment of conversation protocols in terms of landmark expressions and partially specified, concrete protocols.

To summarize, this dissertation extends an existing formal theory of teamwork (Joint Intention Theory) by providing a comprehensive formal semantics of multi-agent communication based on that theory along with support for a wider variety of teams. Thereafter, it presents a domain independent agent programming language called STAPLE with built-in support for teamwork and multi-agent conversations based on these theoretical contributions. The usefulness of STAPLE for programming teams of autonomous agents

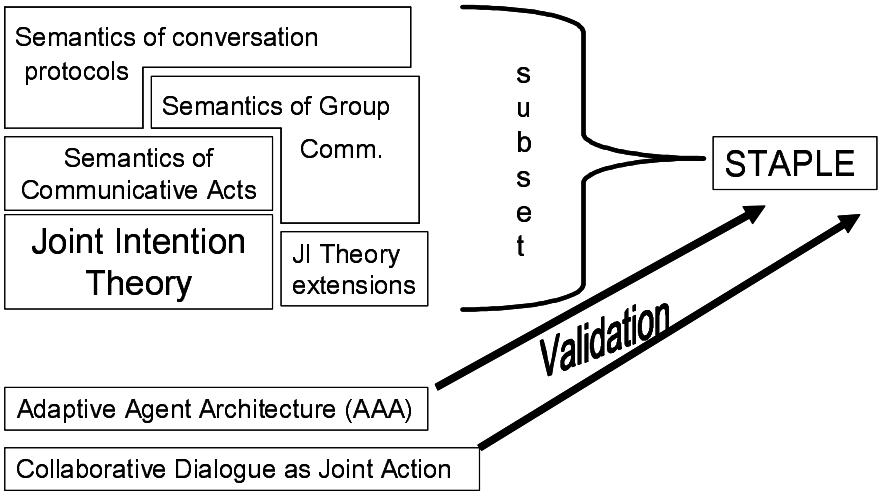


Figure 1.1: Summary of the approach

is demonstrated by showing that correct team and communicative behaviors follow from agent specifications in two different domains without explicitly programming those behaviors in every possible situation. Next, we provide an overview of the research undertaken to address the various problems and their proposed solution mentioned above.

1.5 OVERVIEW

The following sections provide an overview of the space of problems that we investigate as per the above outline. The discussions that follow form the basis of the remaining chapters in this dissertation.

1.5.1 A Formal Semantics for an Agent Communication Language

The philosophy of language argues that the illocutionary effect of a speech act consists of the hearer's recognition of the speaker's communicative intention [100]. A communicative (i.e., speech) act succeeds when the hearer successfully recognizes the speaker's intention and it is satisfied when the hearer successfully acts on the speaker's intention. We must characterize communicative acts as *attempts* because there is a possibility that the act may not succeed. For example, suppose that I sincerely request you to open the door. The goal of my request is that you open the door and the intention of my request is to bring about

mutual belief between us that I want you to open the door. My request is successful if you recognize that I want you to open the door and my request is satisfied if you actually open the door in response to my request. The best I can do is to make my intention known to you and it is up to you whether you actually open the door. If I have reason to believe that you have not properly understood my intention then I may repeat my request, that is, I may attempt again to make my intention known to you. Based on this premise, researchers working on joint intention theory first defined an attempt as having a goal and an intention, and then defined communicative acts in terms of attempt [26, 27, 30]. They argued that communicative acts defined in this manner lead to creation and discharge of commitments. Smith and Cohen [106, 107] took this a step further and showed that the joint intention theory could provide the formal framework for an agent communication language based on speech acts. They also argued that communicative acts performed in certain sequences could lead to the creation and discharge of joint commitments and therefore, one could create and disband teams based on the joint intention theory by using proper communicative acts (assuming that the agent programs respected the formal communication semantics).

This dissertation builds upon and extends the prior work of Smith and Cohen in the following manner: It formally lays down the defeasible rules of communication that are critical for establishing mutual beliefs in this framework, redefines the semantics of communicative acts to be closer to the speech act theory, and formally establishes the properties regarding mutual belief establishment as well as those concerned with the creation and discharge of joint commitments. The redefined communicative act semantics and the accompanying formal results enable first principles reasoning about inter-agent communication based on the joint intention theory; enable analysis of conversation protocols for their correctness; and are critical to the development of an agent programming language where logical specifications of agents are executed directly to get the correct behavior automatically.

Agents in a team rarely use the communicative acts in isolation but rather they use them in the context of a larger conversation. Conversation protocols specify the sequence of communicative acts that constitute a conversation, and therefore, agents must be able

to understand and reason about conversation protocols. Next, we discuss the problem of providing a formal semantics to conversation protocols.

1.5.2 A Formalism for Conversation Protocols

Inter-agent communication is usually an extended conversation to achieve something rather then just a one-shot communication that terminates with the exchange of a single message. If agents were always to reason about communication using first principles then the formal semantics of communicative acts would be sufficient for all agent interactions. However, a large number of interactions in the real world follow explicit or implicit protocols that are either defined by an institution (such as auction protocols used by auction houses), or are generally accepted social conventions. For example, when someone asks you a question, you are expected to either tell the answer, or tell them that you don't know the answer and optionally point them to someone who might know the answer. Constraining a multi-agent conversation to proceed according to the protocol being followed would be far more efficient than full-blown reasoning from first principles every time a message is received or sent. Furthermore, agents that do not have reasoning capabilities need to follow well-defined protocols whose correctness has been proven offline. However, there is no framework for formal semantics of multi-agent conversations in the literature, within the context of teamwork, thereby making it impossible to prove the correctness of a conversation with respect to the task at hand.

Traditionally, conversation protocols are specified as finite state machines in which the transition arcs specify the communicative actions to be used by the various agents involved in a conversation. Protocols are executed by performing these communicative actions and, therefore, the communicative actions have come to be regarded as the central concept around which analyses of protocols are based. However, we believe that it is the states and not the state transitions that are key to the correctness and completeness of a protocol. This dissertation proposes a landmark-based approach for formal analysis of conversation protocols wherein the most important aspect of a conversation protocol is not the set of communicative actions involved in that protocol but the effects or the states that these actions bring about. The basic idea is that since protocols are used to do certain tasks

or to bring about certain state of affairs in the world, one should identify the important landmarks or state of affairs that are brought about by and during the execution of a protocol. Conversation protocols can then be expressed at an abstract level as partially ordered landmarks where each landmark is characterized by the propositions that are true in the state represented by that landmark. The partially ordered landmarks represent a family of protocols. Communicative actions are then the tools to realize concrete protocols from a landmark-based representation. Besides contributing to formal analyses of protocol families, the landmark-based representation facilitates the system's dynamically choosing the most appropriate action to use next in a conversation, allows compact handling of protocol exceptions, and in some cases, even allows short-cutting a protocol *execution* by opportunistically skipping some intermediate landmarks (Chapter 8).

The landmark expressions can be thought of as protocol specifications and they form the basis for design and correctness of concrete protocols. However, it is the concrete protocols that eventually get executed, and therefore, we need a proper formalism for concrete protocols that is suitable for automated reasoning. This dissertation suggests such a formalism using the very definition of conversation protocols as a pattern of communicative actions. It proposes that concrete protocols be represented along with their precondition and goal as action expressions using dynamic logic constructs. These communicative action expressions involve multiple cooperating agents and henceforth will be called joint action expressions. The motivation for joint action expressions also comes by analogy with natural language wherein dialogues are treated as joint actions [46, 31, 28]. The communicative actions in the joint action expression for a protocol must achieve the landmarks of the protocol family of that protocol in the required order. This requirement provides a correctness criterion for concrete conversation protocols – a protocol must be correct with respect to the landmark expression of its protocol family. The proposed formalism also provides a means to represent and reason about protocol compositions. Furthermore, it provides other correctness and completeness criterion for protocols as well as for protocol compositions. Finally, this dissertation explores the possibility of applying existing formal theories of dialogue and teamwork, such as joint intention theory, to protocols represented as joint action expressions. Conversation protocols within this framework can thus be

executed in the same way as any other joint action. Therefore, we can directly execute such protocols by agent programming languages such as STAPLE.

We note that communication between team members invariably requires group communication as discussed next.

1.5.3 A Formal Semantics of Group Communication

Artificial as well as human agents not only interact with individual agents, but they also need to communicate with groups of agents. We humans post messages to mailing lists and notice boards; participate in teleconferences and videoconferences; publish web pages and books; speak in meetings and classrooms; talk on radio and television; and advertise on pamphlets and banners. Agents will be assuming some of these responsibilities from humans, and will therefore need to be able to reason and communicate about group concepts. Moreover, in open multi-agent systems, where agents come and go dynamically, it will become ever more prevalent that agents will not know exactly to whom they are sending information or from whom they are requesting aid. These are compelling reasons to investigate support for group communication in multi-agent systems. It is no surprise, therefore, that a large number of distributed software systems inevitably use some incarnation of broadcasting and multicasting. However, we observe that the major agent communication languages (such as FIPA [41] and KQML [67]) have either no provision or no well-defined semantics for group communication. The problem is to develop a formal semantics for group communication such that group communicative acts reduce to individual communicative acts for the special case of groups containing a single agent.

This dissertation presents a formal semantics for group communicative acts and shows that they satisfy numerous desirable constraints. In particular, we present the semantics of group communicative acts as used in the declarative agent programming language called STAPLE that is discussed later in this dissertation. A consequence of our group communication semantics is that it allows reasoning about communication even if one does not know all the agents that are involved in a communication. Finally, another important consequence is that it allows creation and discharge of persistent teams as described in the next section.

1.5.4 A Formalism for Persistent and Dynamic Teams

A team of agents, though robust to various kinds of failures and uncertainties, is still vulnerable to the sudden unavailability of a teammate. Unavailability of an agent may result from the termination of the agent process due to unexpected conditions, crashing of the machine on which the agent process is running, and network partitioning for unforeseen periods. Robustness from such failures typically requires teams that persist but whose members may change dynamically with time. This is so because new agents can join existing teams to take over the role of incapacitated or unavailable teammates. Furthermore, once a team has recovered from failure, the team needs to continue to exist so as to be able to handle future failures. The importance of persistent and dynamic teams for the purpose of robustness has long been recognized in the multi-agent system literature. For instance, Tambe and Zhang [111] point out the need for a formalism that explains the maintenance of a team identity in the face of changing membership and the persistence of a team beyond specific temporary objectives.

This dissertation departs from our earlier teamwork theory by allowing a team to exist independently of the identity of its members. There are numerous examples of such teams in everyday life. For example, the New York Yankees are a team even if all the players are traded or it is sold to new owners. The earlier work characterized a team in terms of joint commitment between the individuals that originally constituted the team. This dissertation formally defines a notion of team commitment wherein the agents are committed towards "whoever" are members of the team at any time, thus enabling the team to continue even when the team membership changes dynamically.

Further, the earlier characterization of teamwork in terms of joint persistent goal results in the team being dissolved once the jointly committed goal is achieved and is mutually believed. This dissertation presents a semantics for restorative maintenance goals along with a characterization of team commitment based on restorative maintenance goals. This team commitment results in teams that continue beyond one-time joint persistent goals. Whenever a team committed to maintaining a proposition comes to mutually believe that the proposition is not true, it adopts a joint persistent goal to achieve that proposition.

This joint persistent goal is discharged once the required proposition is achieved and mutually believed but the original maintenance commitment remains valid and the team continues to exist.

We believe that the concept of team commitment and maintenance goals introduced in this dissertation provides a formalism that can be gainfully employed in building robust agent teams. In fact, a team specification using these concepts has been implemented in a multi-agent system architecture thereby making systems based on this architecture robust to certain kinds of broker failures. A discussion on this fault-tolerant architecture called the Adaptive Agent Architecture (AAA) follows. This agent architecture provides a validation criterion that must be met by the agent programming language proposed later in this dissertation.

1.5.5 A Fault-tolerant Multi-agent System Architecture

Multi-agent systems are prone to the failures that can occur in any distributed software system. Bugs and improperly handled exceptions in the agent program or in the supporting environment, machine crashes, network partitioning, and numerous other hardware and software faults can make agents unavailable suddenly for unforeseen periods. The traditional distributed systems literature provides various fault-tolerance techniques to recover from these failures. This dissertation argues that most of these techniques are meant for specific failure situations and that they require special infrastructural support. For example, the techniques of hot backups, object group replication, virtual synchrony, and N-version voting need specific mechanisms for communication and synchronization among the replicas. It may not be possible to use these techniques in multi-agent systems without extensive modifications to the underlying agent infrastructure. On the other hand, a technique based on a multi-agent system concept may be amenable to implementation with minimal modifications, for example, by adding a plan to the plan library of agents.

Multi-agent systems often require brokers (or middle agents) for many reasons including: accepting requests, locating capable agents, routing requests and responses, sharing of information, managing the system, registering agent capabilities, and for various other facilitation tasks [116]. However, brokered systems are brittle because the facilitator is a

single point of failure. Our experience with Quickset [24], a multi-agent system based on the facilitated Open Agent Architecture [23, 72], reinforces the need for an agent architecture that can recover quickly from broker failures.

We observe that, in a large multi-agent system, there will typically be more than one middle agent and that these middle agents may be able to substitute for one another when needed. Therefore, JI theory argues that, if these middle agents form a team with appropriate joint commitments, then they will compensate for any middle agent that becomes unavailable. As a result, the multi-agent system will continue to work as long as there is at least one middle agent remaining in the broker team. However, the performance may degrade as a result of having fewer middle agents in the system. We further hypothesize that a broker team with a commitment to maintain a specified minimum number of brokers in the broker team will attempt to restore the population of the team. This may result in a recovered system with similar performance as the original system.

This dissertation presents the Adaptive Agent Architecture (AAA), a robust multi-brokered multi-agent system architecture, to show the feasibility of our approach. The AAA uses teamwork (1) to recover a multi-agent system broker failures, and (2) to maintain a specified minimum number of functional brokers in the system even when some of the brokers become inaccessible. The AAA agent library has been developed in Java and it provides an agent shell for developing AAA agents. The library also provides a facilitator agent that serves both as a broker and a matchmaker. Henceforth, in this dissertation, we will refer to the AAA facilitator as the AAA broker. The AAA brokers can be interconnected with each other and the agent library supports both facilitated and direct inter-agent communication. The AAA agents advertise their capabilities as well as an address for connection requests with a broker during registration. TCP/IP is used for network transport and the TCP mechanisms and timeouts are used for detection of connection failures. The brokers as well as other agents can dynamically enter and leave AAA-based multi-agent systems. The AAA brokers form a team for the purpose of fault-tolerance and they share knowledge about who is connected to whom with the team members.

The AAA brokers form a persistent broker team when they register with each other.

This broker team establishes a *team intention* and a *team maintenance goal* for the purpose of fault-tolerance as specified in the following mission statements.

AAA Mission Statement 1: Whenever an agent registers with the broker team, the brokers have a team intention of connecting with that agent, if it ever disconnects, as long as it remains registered with the team.

AAA Mission Statement 2: The AAA broker team has a team maintenance goal of having at least N brokers in the team at all times where N is specified during the team formation.

The design of the AAA brokers implements the specification of teamwork that follows from the mission statements. Using the mission statements, along with other logical properties of the AAA, we establish the commitments of the brokers in the team. These commitments result in fault tolerant behavior when the brokers act rationally and take appropriate actions to honor them. This dissertation presents the logical characterization of brokers in the AAA and formally establishes the design specifications that are implemented in the AAA brokers.

The fault-tolerance of AAA brokers demonstrates the usefulness of teamwork. However, there is no straightforward way to change that specification to get a different team behavior without having to re-implement the new behavior predicted by the changed specification. Furthermore, there is no way to guarantee that all predications that follow from the specification have been accounted for, leaving open the possibility that their might exist (currently unknown) situations in which the team behavior of AAA brokers will cease to function correctly. This drawback provides a strong motivation for a declarative agent programming language whose interpreter directly executes the team specifications. Furthermore, the AAA research provides a validation criterion for this programming language – brokers written in this language must be able to replicate the behavior of AAA brokers upon providing them with the declarative specification of that behavior. A brief discussion on this programming language called STAPLE follows.

1.5.6 STAPLE: A Declarative Agent Programming Language

Many implementations of teamwork use joint intention theory only for specification purposes and thereby hard-code the team behavior predicted by the joint intention theory [52, 63]. However, in these cases implementing a different team behavior requires reprogramming the new behavior according to the theory. On the other hand, the benefits of directly interpreting a team specification are apparent from the experiences of the STEAM researchers (Tambe and colleagues [109, 110, 111]). STEAM (**S**hell for **TEAM**work) is a set of SOAR [69] rules that attempts to execute joint commitments faithfully. Tambe and colleagues have demonstrated successful reuse of the same STEAM code base for a number of applications. We take this concept of directly executing team specifications to the next step by offering interpretation of agent specifications in a logic that is used for formal specification of the joint intention theory.

STAPLE (*Social and Team Agents Programming Language*) is a multi-agent programming language with built in support for teamwork and multi-agent conversations. It enables programming multi-agent systems by directly executing the specification of agents in a subset of modal logic, dynamic logic of actions, and temporal logic along with abstractions from the joint intention theory as well as from a formal semantics of multi-agent conversations based on the joint intention theory. As such, STAPLE is a domain independent agent programming language that is formally connected with a logical theory of agency, and whose constructs, including the communication primitives, have a well-founded formal semantics. The benefits of this approach include the ability to modify agent and team behaviors just by modifying a single sentence in the logical language, and a potential for verification – for instance, one may be able to predict a team behavior offline using the logical specification of agents involved and verify it by running the actual system.

Beliefs, goals, commitments, and intentions are represented explicitly in STAPLE. Actions are required to have a logical representation that can be used for reasoning, and plans as well as conversation protocols are treated as complex action expressions consisting of action sequences, non-deterministic OR, concurrent actions, repetitions, and test actions.

The axioms of rational behavior are specified as rules, which agent programmers can override. STAPLE agents can have multiple simultaneous commitments and intentions, with a notion of importance used to order everything from commitments and intentions to plans and rules. The syntax of STAPLE is presently an extension of the usual Prolog syntax with the exception that certain constructs such as primitive actions and plans can also be written in Java. Most of the STAPLE interpreter, including a multi-threaded Prolog subsystem, has been implemented in Java but a few components such as the belief reasoner and the default rules have been written in Prolog.

The approach of directly executing formal specifications offers a potential for verification as mentioned earlier, and also has the benefits of code reuse at a very high level by virtue of the ability to alter the behavior of a system by modifications to its formal specification. The ability to modify team behavior at a high level enables quick prototyping of teamwork. The support for persistent teams in STAPLE allows interesting team behaviors to be implemented using few logical sentences. This dissertation implements and discusses two such examples. First, the fault-tolerance specification of AAA brokers is declaratively given to brokers written in STAPLE and it is shown that these brokers exhibit the same fault-tolerance behavior as that of AAA brokers by virtue of interpreting the declarative logical sentence. Second, STAPLE agents are shown to exhibit correct team and communicative behavior in a simulated game that involves human-agent collaboration. Furthermore, direct execution of specifications may enable one either to prove that the operating specifications of a system are not violated (i.e., that the system always behaves according to a required set of properties), or to identify the properties of the system that cannot be proven to obey the specification. It also offers the advantages of dynamic program synthesis and execution: Legacy code accompanied by its description in the formal language can be treated as an "action" by an agent. The agent then reasons about those actions using their formal description, decides upon and executes the best course of action(s) at any given time, and even switches between intentions (and therefore, between the legacy codes being executed) as the priorities of the agent change. One can argue that this approach not only lets one "agentify" any system but also prove that the modified system does in fact have the required agent-like properties. For instance, it

should be possible to treat a non-team aware agent as part of a team by using a proxy agent that reasons using the formal representation of actions of the non-team aware agent, and be able to prove formally the conditions under which the non-team aware agent does in fact appear to behave as a team member. The concept of team-proxies has been successfully employed by other researchers such as Pynadath et. al. [88] and the STAPLE approach would not only justify those prior efforts but also provide a theoretically sound way of achieving similar end. These arguments are powerful motivations that make the direct execution approach in STAPLE worth pursuing.

The STAPLE interpreter needs logical reasoning capabilities along with the ability to handle procedural tasks such as control flow and stack manipulation. Logic programming languages such as Prolog are good for logical reasoning but procedural tasks can quickly get quite complex and unwieldy in such languages. It was quite clear from our early experience [62] with STAPLE that completely implementing a STAPLE interpreter in Prolog was not a viable option because a large portion of the interpreter dealt with procedural control, and the commercially available multi-threaded Prolog implementations were too buggy and quickly broke down when used with STAPLE. Similarly, imperative languages such as Java are good for procedural tasks but are ill-suited for logical reasoning. As such, we take a hybrid approach for the current STAPLE development by choosing to use both Prolog and Java, and using each language for tasks that they do best – Java is used for procedural control and Prolog is used for logical reasoning. This dissertation presents an actual implementation of a STAPLE interpreter. A formal operational semantics of STAPLE based on joint intention theory (that forms the basis of the implemented interpreter) is presented in the appendix.

The usefulness of STAPLE for programming teams of autonomous agents is demonstrated by showing that correct team and communicative behavior follows from the specification of an agent in STAPLE without having to program the team behavior explicitly in every possible situation by (1) directly executing the AAA fault-tolerance specification to get the same behavior as that of AAA based multi-agent systems, and (2) showing that STAPLE agents exhibit conversational behavior by virtue of reasoning about communicative acts and teamwork using first principles. In short, this dissertation demonstrates the

feasibility of a logic-based interpreted language for programming teams of autonomous agents that automatically exhibit correct team and communicative behavior, and that can handle certain unforeseen situations by virtue of first principles reasoning over a formal theory of teamwork and multi-agent communication.

1.6 SUMMARY AND OUTLINE

To summarize, this dissertation demonstrates the feasibility of a logic-based declarative language for programming teams of autonomous agents that exhibit correct team and communicative behavior without having to program that behavior explicitly. It extends an existing formal theory of teamwork (Joint Intention Theory) by providing a comprehensive formal semantics of multi-agent communication based on that theory along with support for a wider variety of teams. Thereafter, it presents a domain independent agent programming language called STAPLE with built-in support for teamwork and multi-agent conversations based on these theoretical contributions. STAPLE agents are programmed using a subset of modal logic, dynamic logic of actions, and temporal logic along with teamwork constructs and communication primitives that have a well-founded formal semantics. The usefulness of STAPLE for programming teams of autonomous agents is demonstrated by showing that correct team and communicative behaviors follow from agent specifications in two different domains without having to explicitly program those behaviors in every possible situation.

The remainder of this dissertation is organized according to the approach outlined earlier for establishing the hypothesis that it is possible to design and implemented a declarative agent programming language with built-in support for teamwork and communication. Chapter 2 provides the formal background by introducing the logic of joint intentions and the semantics of communicative acts. Chapter 3 introduces STAPLE by means of concrete examples of agent programs written in STAPLE, and Chapter 4 presents the implementation of one possible STAPLE interpreter that is used to execute the examples and test cases in this dissertation. Chapter 5 presents a generalized theory of teamwork and communication to support groups. The Adaptive Agent Architecture (AAA) is discussed in

Chapter 6, and Chapter 7 presents the implementation of AAA fault tolerant behavior in STAPLE, thereby showing that STAPLE validates the above hypothesis. Chapter 8 discusses multi-agent conversations and shows how STAPLE executes fully specified protocols. This dissertation concludes with a discussion of related work in Chapter 9, and future work and summary in Chapter 10. A formal definition of the programming language STAPLE and its operational semantics based on joint intention theory is presented in the appendix.

Chapter 2

A Logic of Teamwork and Communication

Joint intention theory provides a formal model of teamwork as well as a framework for formal semantics of multi-agent communication. This theory is expressed in a modal language that has the usual connectives of a first order logic with equality and operators for propositional attitudes and event sequences. The primitive mental states in this theory are an agent's beliefs and goals, expressed as (BEL x p) and (GOAL x p) respectively, where x is an agent and p is a proposition that follows from x's beliefs or goals. BEL has Kripke's weak S5 semantics and GOAL has system K semantics. Knowledge (KNOW x p) is defined as true belief in the standard manner. The goals of an agent are constrained to be compatible with its beliefs. An agent is competent[1] with respect to these primitive mental states. (AGT x a) says that x is the only agent for the sequence of events represented by a. Temporal properties are expressed in a linear time temporal logic. $\Diamond p$ says that the proposition p will eventually be true and $\Box p$, defined as $\neg\Diamond\neg p$, says that p will always be true. (HAPPENS a) and (DONE a) say that a sequence of actions described by the action expression a will happen next or has just happened, respectively. (HAPPENS x a) and (DONE x a) also specify the agent for the action sequence that is going to happen or has just happened. EARLIER, BEFORE, AFTER, UNTIL, and DOING are defined using HAPPENS and DONE. An action expression is built from variables ranging over

[1]From Cohen & Levesque [25], an agent is competent with respect to a proposition if that proposition is true whenever the agent believes it to be so, that is, (COMPETENT x p) $\triangleq$ (BEL x p) $\supset$ (KNOW x p).

sequences of events using constructs of dynamic logic: $a;b$ is action sequence, $a|b$ is non-deterministic choice, $a||b$ represents concurrent actions, $a*$ is indefinite repetition, and $p\models$ is a test action. Details of this modal language and its model theory can be found in Cohen & Levesque [25].

We first review the syntax and the model theory of basic concepts in this logical framework from [70, 106, 25] followed by a brief overview of the joint intention theory. Thereafter, we establish some useful results that are needed in later chapters, provide a formal semantics of communicative acts, and show that these communicative acts can be used to create and discharge teams as per this theory.

2.1 SYNTAX

The syntax of the logical language behind the joint intention theory is specified as follows.

⟨Action-var⟩ =:: a,b,a_1,a_2,...,b_1,b_2,...,e,e_1,e_2,...

⟨Agent-var⟩ =:: x,y,x_1,x_2,...,y_1,y_2,...

⟨Group-var⟩ =:: τ,α, τ_1, τ_2,..., α_1, α_2,...

⟨Regular-var⟩ =:: i,j,i_1,i_2,...,j_1,j_2,...

⟨Variable⟩ =:: ⟨Action-var⟩ | ⟨Agent-var⟩ | ⟨Group-var⟩ | ⟨Regular-var⟩

⟨Pred⟩ =:: (⟨Pred-symbol⟩⟨Variable$_1$⟩ ... ⟨Variable$_n$⟩)

⟨Time-proposition⟩ =:: ⟨Numeral⟩

⟨Action-expression⟩ =:: ⟨Action-var⟩| one of the following:

⟨Action-expression⟩;⟨Action-expression⟩ */*Sequential Action*/*

⟨Action-expression⟩|⟨Action-expression⟩ */*Non-Deterministic Choice Action*/*

⟨Action-expression⟩||⟨Action-expression⟩ */*Concurrent Action*/*

⟨Wff⟩ $\models$ */*Test Action*/*

⟨Action-expression⟩* */*Iterative Action*/*

⟨Wff⟩ =:: ⟨Pred⟩ | ¬⟨Wff⟩ | ⟨Wff⟩ ∨ ⟨Wff⟩ |

∃⟨Variable⟩⟨Wff⟩ where ⟨Variable⟩ is free in ⟨Wff⟩ | one of the following:

⟨Variable⟩ = ⟨Variable⟩,

(HAPPENS ⟨Action-expression⟩), /∗Action-expression *happens next */*

(DONE ⟨Action-expression⟩), /∗Action-expression *has just happened* */

(AGT ⟨Agent-var⟩⟨Action-var⟩), /* Agent-var is the only agent of Action-var */

(BEL ⟨Agent-var⟩⟨Wff⟩), /∗Wff *follows from* Agent-var*'s beliefs* */

(GOAL ⟨Agent-var⟩⟨Wff⟩), /∗Wff *follows from* Agent-var*'s goals* */

⟨Time-Proposition⟩,

⟨Action-var ⟩ ⩽ ⟨Action-var⟩

The syntax (and the semantics) of any new concepts will be presented as and when they are introduced.

2.2 MODEL THEORY

The model theory of the above language interpreted by abstract STAPLE interpreter is based on a possible-worlds semantics where worlds are modeled as a linear sequence of primitive event types. Formulas are evaluated with respect to some possible course of events (*possible worlds*) and an index into that possible world. The possible worlds can be related to each other via belief and goal accessibility relations.

A model $\mathbb{M}$ is a structure $\langle \Theta, \mathcal{P}, \mathcal{E}, \text{Agt}, \mathcal{T}, \mathcal{B}, \mathcal{G}, \Phi \rangle$ where Θ is a set of things, $\mathcal{P}$ is a set of people, $\mathcal{E}$ is a set of primitive event types, $\text{Agt} \in [\mathcal{E} \rightarrow \mathcal{P}]$ specifies the agent of an event, $\mathcal{T} \subseteq [\mathbb{Z} \rightarrow \mathcal{E}]$ is a set of possible courses of events (worlds), $\mathcal{B} \subseteq \mathcal{T} \times \mathcal{P} \times \mathbb{Z} \times \mathcal{T}$ is the belief accessibility relation, $\mathcal{G} \subseteq \mathcal{T} \times \mathcal{P} \times \mathbb{Z} \times \mathcal{T}$ is the goal accessibility relation, and Φ interprets predicates. The domain of quantification ($\mathcal{D}$) is given by $\mathcal{D} = \Theta \cup \mathcal{P} \cup \mathcal{E}^*$ meaning that one can quantify over things, people, and sequences of primitive event types. The predicate interpreter is specified by $\Phi \subseteq [\text{Pred}^k \times \mathcal{T} \times \mathbb{Z} \times \mathcal{D}^k]$, where Pred represents predicates. AGT specifies the partial agents of a sequence of events and is given by $AGT \subseteq \mathcal{E}^* \times \mathcal{P}$ where $\text{x} \in \text{A}GT[e_1,\ldots,e_n]$ iff $\exists i$ such that $\text{x} = \text{Agt}(e_i)$. The notions of satisfaction and validity in this model are defined next.

2.2.1 Satisfaction and Validity

The satisfaction and validity of a well-formed formula (wff) is defined as follows. Let $\mathbb{M}$ be a model, σ be a sequence of events (i.e., a world), ν be a variable assignment, and

n $\in \mathbb{Z}$ be a temporal index into the world. Let $\nu \in$ [Vars$\rightarrow \mathcal{D}$]. Let ν_x^d be the function that yields d for x and is the same as ν everywhere else. Let n$[\![a]\!]$m denote that a occurs between time points n and m.

Satisfaction: A wff α is satisfiable if there is at least one model $\mathbb{M}$, world σ, index n, and value assignment ν such that $\mathbb{M}$,σ,ν,n $\models \alpha$.

Validity: A wff α is valid (denoted by $\models \alpha$) iff for every model $\mathbb{M}$, world σ, index n, and value assignment ν, we have $\mathbb{M}$,σ,ν,n $\models \alpha$.

Interpretation of predicates depends on the world σ and index n. For a k-place predicate P, we have

$\mathbb{M}, \sigma, \nu$,n $\models$ P(x$_1$,...,x$_k$) iff $\langle \nu(\mathrm{x}_1) \ldots \nu(\mathrm{x}_k) \rangle \models \Phi[\mathcal{P}, \sigma$,n]

The satisfaction of the basic operators (negation, disjunction, existential quantifier, and equality) and propositions is defined as follows.

1. $\mathbb{M}, \sigma, \nu$,n $\models \neg\alpha$ iff $\mathbb{M}$,σ,ν,n $\models \alpha$
2. $\mathbb{M}, \sigma, \nu$,n $\models \alpha \vee \beta$ iff $\mathbb{M}$,σ,ν,n $\models \alpha$ or $\mathbb{M}$,σ,ν,n $\models \beta$
3. $\mathbb{M}, \sigma, \nu$,n $\models \exists$x α iff $\mathbb{M}$,σ,ν_x^d,n $\models \alpha$ for some d $\in \mathcal{D}$
4. $\mathbb{M}, \sigma, \nu$,n $\models$ (x1 = x2) iff ν(x1) = ν(x2)
5. $\mathbb{M}, \sigma, \nu$,n $\models \langle$Time-proposition$\rangle$ iff $\nu(\langle$Time-proposition$\rangle)$ = n
6. $\mathbb{M}, \sigma, \nu$,n $\models$ (AGT x e) iff $AGT[\nu(\mathrm{e})] = \{\nu(\mathrm{x})\}$ where AGT specifies the only agent of e.
7. $\mathbb{M}, \sigma, \nu$,n $\models$ (e$_1 \leqslant$ e$_2$) iff ν(e$_1$) is an initial subsequence of ν(e$_2$)

Next, we define satisfaction of the primary modal operators in this logic.

2.2.2 Satisfaction of Primary Modal Operators

The satisfaction of the primary modal operators is defined as follows.

1. (BEL x p) says that p follows from the agent's beliefs iff p is true in all possible worlds accessible via $\mathcal{B}$ at index n. Formally,
   $\mathbb{M}, \sigma, \nu$,n $\models$ (BEL x p) iff $\forall\sigma^*$ such that $\langle\sigma$,n$\rangle\mathcal{B}$[v(x)]σ^*, $\mathbb{M}$,σ^*,ν,n $\models p$

2. (GOAL x p) says that p follows from the agent's goals iff p is true in all possible words accessible via $\mathcal{G}$, at index n. Formally,
   $\mathbb{M}, \sigma,\nu,$n $\models$ (GOAL x p) iff $\forall\sigma^*$ such that $\langle\sigma,\text{n}\rangle\mathcal{G}[\text{v(x)}]\sigma^*$, $\mathbb{M},\sigma^*,\nu,$n $\models p$

3. (HAPPENS a) says that a describes a sequence of events that happens next. Formally,
   $\mathbb{M}, \sigma,\nu,$n $\models$ (HAPPENS a) iff $\exists$m, m$\geqslant$n, such that $\mathbb{M},\sigma,\nu,\text{n}[\![a]\!]\text{m}$

4. (DONE a) says that a describes a sequence of events that just happened. Formally,
   $\mathbb{M},\sigma,\nu,$n $\models$ (DONE a) iff $\exists$m, m$\leqslant$n, such that $\mathbb{M},\sigma,\nu,\text{m}[\![a]\!]\text{n}$

(HAPPENS x a), (DONE x a), (HAPPENS x y a) and (DONE x y a) also specify the agent(s) for the sequence of events that constitute a. For example, if a is a singleton event, then

$$\text{(DONE x } a\text{)} \triangleq \text{(DONE } a\text{)} \wedge \text{(x = Agt}(a)\text{)}$$

Next, we define what it means for events and actions to occur in this model.

2.2.3 Occurrence of Events and Actions

The occurrence of events and actions in this model is specified in the following manner.

Event variable e denotes a sequence of events of length m starting at time n. In other words,

$\mathbb{M}, \sigma,\nu,\text{n}[\![e]\!]\text{n+m}$ iff $\nu(\text{e}) = e_1e_2\ldots e_m$ and $\sigma(\text{n+}i) = e_i$, $1\leqslant i \leqslant$m.

1. *Action Sequence:* The sequence $a;b$ says that the action described by a and then that described by b occurs. Formally,
   $\mathbb{M}, \sigma,\nu,\text{n}[\![a;b]\!]\text{m}$ iff $\exists$k, n$\leqslant$k$\leqslant$m, such that $\mathbb{M},\sigma,\nu,\text{n}[\![a]\!]\text{k}$ and $\mathbb{M},\sigma,\nu,\text{k}[\![b]\!]\text{m}$

2. *Non Deterministic Choice:* The non-deterministic OR $a|b$ says that either the action described by a or that described by b occurs. Formally,
   $\mathbb{M}, \sigma,\nu,\text{n}[\![a|b]\!]\text{m}$ iff $\mathbb{M},\sigma,\nu,\text{n}[\![a]\!]\text{m}$ or $\mathbb{M},\sigma,\nu,\text{n}[\![b]\!]\text{m}$

3. *Concurrent Action:* The concurrent action $a||b$ says that both the actions described by a and b occur. Formally,

$\mathbb{M}, \sigma, \nu, \mathrm{n}[\![a||b]\!]\mathrm{m}$ iff $\exists \mathrm{m}_1$, m_2 where $\mathrm{n} \leqslant \mathrm{m}_1 \leqslant \mathrm{m}$ and $\mathrm{n} \leqslant \mathrm{m}_2 \leqslant \mathrm{m}$ such that$\mathbb{M}, \sigma, \nu, \mathrm{n}[\![a]\!]\mathrm{m}_1$ and $\mathbb{M}, \sigma, \nu, \mathrm{n}[\![b]\!]\mathrm{m}_2$

4. *Test Action:* The test action $p?$ occurs if p holds or blocks (fails) when p is false. Formally,

   $\mathbb{M}, \sigma, \nu, \mathrm{n}[\![p?]\!]\mathrm{m}$ iff $\mathbb{M}, \sigma, \nu, \mathrm{n} \models p$

5. *Indefinite repetition:* The indefinite repetition a^* describes the iterative action $a;a^*$ Formally,

   $\mathbb{M}, \sigma, \nu, \mathrm{n}[\![a^*]\!]\mathrm{m}$ iff $\exists \mathrm{n}_1, \ldots, \mathrm{n}_k$ where $\mathrm{n}_1 = \mathrm{n}$ and $\mathrm{n}_k = \mathrm{m}$ and for every i such that $1 \leqslant i \leqslant \mathrm{k}$, $\mathbb{M}, \sigma, \nu, \mathrm{n}_i\, [\![a]\!]\mathrm{n}_{i+1}$

The basic logical operators, the primary modal operators, and the action expression operators can be combined to define several useful and more commonly recognized concepts. The next section shows examples of some of these common concepts.

2.2.4 Abbreviations

1. *Empty sequence:* As a test action, the empty sequence nil always succeeds. Formally,

   $\mathrm{nil} \triangleq (\forall\, \mathrm{x}\ (\mathrm{x} = \mathrm{x}))?$

2. *Event subsequence*: As an event sequence, NIL is a subsequence of every other event. Formally,

   $a = \mathrm{NIL} \triangleq \forall b\ (a \leqslant b)$

3. *Conditional action & while loop:* The conditional action and the while loop are defined using action expressions as follows:

   [IF p THEN a ELSE b] $\triangleq (p?;a)|(\neg p;b)$

   [WHILE p DO a] $\triangleq (p?;a)^*|\neg p?$

4. *Eventually:* Eventually p denoted by $(\Diamond p)$ is true in a given possible world if there is some sequence of events after which p will hold, that is if p is true at some point in the future. Formally,

$\Diamond p \triangleq \exists$x (HAPPENS x;$p$?)

5. *Always:* Always p denoted by ($\Box$p) means that p is hence forth true in the course of events. Formally,

   $\Box$p $\triangleq \neg\Diamond\neg p$

6. *Before & After*[2]*:* The temporal relation (BEFORE a p) says that proposition p was true before action a was done. Similarly, (AFTER a p) says that p will be true after a happens. Formally,

   (BEFORE a p) $\triangleq$ (DONE p?;a)

   (AFTER a p) $\triangleq$ (HAPPENS a;p?)

7. *Until:* The temporal relation (UNTIL p q) says that q remains true at least until p becomes true, that is, for every event c such that q becomes false right after c happens, there exists an event a which is an initial subsequence of c such that p becomes true right after a happens. Formally,

   (UNTIL p q) $\triangleq \forall c$ (HAPPENS q?;c;$\neg q$?) $\supset \exists a$ ($a \leqslant c$) $\wedge$ (HAPPENS $\neg p$?;a;p?)

8. *Earlier:* The temporal relation (EARLIER p) says that p was true in the past but it is currently not true. Formally,

   (EARLIER p) $\triangleq \neg p \wedge \exists e$(DONE p?;e;$\neg p$?)

9. *Doing:* An agent is DOING an action a if the agent is the actor of that action and if one of the following is true (i) a has just been done, or (ii) a is going to happen next (i.e., a is just starting), or (iii) there exists some initial subsequence of a that has just been done but a has not just been done (i.e., a has started but not yet completed). Formally,

   (DOING a) $\triangleq$ (DONE a) $\vee$ (HAPPENS a) $\vee$ [$\exists e$ ($e \leqslant a$) $\wedge$ (DONE e) $\wedge \neg$(DONE a)]

[2]In this dissertation, we use the definitions of BEFORE and AFTER from [26].

10. *Know:* Knowledge is defined as true belief in the usual manner. Formally,

    (KNOW x p) $\triangleq p \wedge$ (BEL x p)

Next, we discuss the constraints imposed on the above model to account for the basic axioms involving modal operators in this logic.

2.2.5 Constraints on the Model

Realism: An agent cannot have a goal that is incompatible with its belief, for example, that it believes to be impossible. Therefore, $\mathcal{G} \subseteq \mathcal{B}$ (i.e., the goal accessible worlds are a subset of the belief accessible worlds). In other words,

$\models$ (BEL x p) $\supset$ (GOAL x p)

Note that this is true relative to a given time as agents can want to change things in the future.

Necessitation: If p is a theorem (i.e., is valid), then it follows from the agent's beliefs at all times. In other words,

if $\models p$, then $\models$ (BEL x $\Box p$)

In this model, the belief accessibility relation $\mathcal{B}$ is constrained to be Euclidean, transitive, and serial. Therefore, BEL has weak-S5 (i.e., KD45) semantics and that gives us the following belief axioms.

1. *Distribution Axiom* (or Axiom K): It follows from the possible worlds approach itself regardless of any constraints we might place on the accessibility relation. It says that if an agent believes p and if it believes that p implies q, then it also believes q.

   (BEL x p) $\wedge$ (BEL x ($p \supset q$)) $\supset$ (BEL x q)

   It is also written in the following equivalent form:

   (BEL x ($p \supset q$)) $\supset$ (BEL x p) $\supset$ (BEL x q)

2. *Consistency Axiom* (or Axiom D): It follows when the accessibility relation is constrained to be *serial* (that is there are no trap states in which no world is accessible). It says that, if an agent believes p, then it cannot also believe $\neg p$, that is,

   (BEL x p) $\supset \neg$(BEL x $\neg p$)

3. *Positive Introspection Axiom* (or Axiom 4): It follows when the accessibility relation is constrained to be *transitive*. It says that, if an agent believes p, then it believes that it believes p.

   (BEL x p) $\supset$ (BEL x (BEL x p))

4. *Negative Introspection Axiom* (or Axiom 5): It follows if the accessibility relation is constrained to be *Euclidean*. It says that an agent is aware of what it does not believe.

   $\neg$(BEL x p) $\supset$ (BEL x $\neg$(BEL x p))

   We also assume that the converse of the positive introspection axiom holds in our model.

5. *Converse of positive introspection axiom:* It says that if an agent believes that it believes p, then it does in fact believe p.

   (BEL x (BEL x p)) $\supset$ (BEL x p)

In this model, the goal accessibility relation $\mathcal{G}$ is constrained to be serial. Therefore, GOAL has system-K (i.e., KD) semantics where axioms K and D are similar to those for BEL. Note that because of the consistency axioms (Axiom D), an agent's beliefs and goals must be separately consistent with each other.

We now provide an overview of Joint Intention theory using the logic framework described so far and establish some of the important results that will be needed subsequently.

2.3 JOINT INTENTION THEORY

Joint Intention theory is expressed in the modal language whose syntax and model theory were presented above. Here, we first discuss the definitions of individual commitment and intention in this theory followed by definitions of joint commitment, joint intention, and related concepts.

2.3.1 Individual Commitment and Intention

In joint intention theory, the notion of an agent's commitment to achieving some state in the world is expressed as a persistent goal or PGOAL [25].

Definition 2.1. Persistent Goal

$$(\text{PGOAL x } p\ q) \triangleq (\text{BEL x } \neg p) \wedge (\text{GOAL x } \Diamond p) \wedge (\text{UNTIL } [(\text{BEL x } p) \vee (\text{BEL x } \Box\neg p) \vee (\text{BEL x } \neg q)](\text{GOAL x } \Diamond p))$$

An agent x has a persistent goal (PGOAL x p q) if x wants p to become true and cannot give up the goal that p is true in the future, at least until it believes that p is accomplished, or is impossible, or is irrelevant (i.e., the relativizing condition q is untrue). The usual convention is to omit the relativizing condition q if it is always the constant *true*. This dissertation assumes that agents are competent with respect to their individual commitments, that is, if an agent believes that it has a PGOAL, then it does in fact have that PGOAL. A consequence of this assumption is that an agent *knows* that it cannot drop its PGOAL at least until one of the conditions in the UNTIL clause becomes true. Formally,

Assumption 2.1. *Agents are competent with respect to their commitments.*

$$\models (\text{BEL x } (\text{PGOAL x } p)) \supset (\text{KNOW x } (\text{PGOAL x } p)) \supset (\text{PGOAL x } p)$$

An intention to do an action a relative to q is represented by (INTEND x a q) and it is defined as a persistent goal in which the agent x is committed to performing the action a believing throughout that it is doing the action.

Definition 2.2. Intention

$$(\text{INTEND x } a\ q) \triangleq (\text{PGOAL x } (\text{DONE x } (\text{BEL x } (\text{DOING x } a))?; a)\ q)$$

The consequences of intending different kinds of action expressions have been analyzed in detail in [29, 25, 28]. For instance, (INTEND x $p?$ q) reduces to (PGOAL x (KNOW x p) q), that is, an intention to do a test action $p?$ results in the agent being committed to coming to know p (and it does not know p now). However, the agent is not committed to bringing about p itself.

Stepwise Execution Strategy: Even though an agent may know that it is executing a sequential action, the definition of intention does not require it to know where it is in the action sequence. As such, an agent who intends to do a sequential action does not necessarily intend to perform the first step in the sequence, even relative to the larger intention. It is consistent with the definition of intention that an agent can intend to do the sequence without expecting to know when the first part of that sequence is over. Moreover, the agent need not intend to do the remainder of the sequence either because it might not know when the first part has been completed and it might not know that it is doing the second part. Therefore, it is possible to execute a sequence of actions without knowingly executing the individual steps because an agent may not know when sub-actions start and stop. However, we can require an agent to believe after each step both that the step was just done and that she is doing the remainder. This execution strategy is called *stepwise execution* [29]. In the stepwise execution of an action sequence, each step becomes a contextual action because it must be performed in a context where the agent has certain beliefs. It follows that, if an agent intends to do a sequential action in a stepwise fashion, then the agent also intends to do each of the steps relative to the larger intention.

Assuming stepwise execution, it has been shown that intending an action sequence $a;b$, that is, (INTEND x $a;b$ q) first results in the agent intending to do the action a. However, the agent does not intend to do b in the beginning, rather, it intends to do (DONE x a)?;b meaning that the agent intends to do b in the context of having done a but it does not intend to do b in isolation. However, once the agent believes that it has just done a while executing $a;b$ then it intends to do b. Furthermore, the agent has the intention to do $a;b$ all along so that the agent is committed to retrying $a;b$ in case of any failure. Therefore, intentions to do a complex action expression (such as action sequence) result in the intentions at the right times to do the component actions.

Similarly, intention to do an iterative action, that is, (INTEND x (WHILE p DO a) q) implies that the agent is committed to executing the action a until the condition p is false. As such, when the agent believes that it has done the iterative action, it will believe that the condition p is false. Intention to do a conditional action, that is,

(INTEND x (IF p THEN a ELSE b) q) implies that the agent cannot be ignorant about the condition p forever. This is because the agent cannot believe that it is about to do a conditional without believing that the condition is true or believing that the condition is false. However, an agent need not know the truth value of a condition on an if-then-else action if the two branches share an initial sequence of events. Only at the point at which those execution paths diverge will it be necessary for the agent to have a belief about the truth of the past condition. These results from the literature on intending action expressions are summarized in Table 2.1. Intention is defined in terms of PGOAL and

Table 2.1: Theorems on Intending Action Expressions

Action Sequences

$\models$ (INTEND x $a_1;a_2;\ldots;a_n$ q) $\wedge$ $\neg$(BEL x (EARLIER (DONE a_1)))
$\supset$ (INTEND x a_1 (INTEND x $a_1;a_2;\ldots;a_n$ q))

$\models$ (INTEND x $a_1;a_2;\ldots;a_n$ q) $\wedge$ (BEL x (DONE $a_1;a_2;\ldots;a_i$))
$\supset$ (INTEND x $a_{i+1};\ldots;a_n$ (INTEND x $a_1;a_2;\ldots;a_n$ q))

Repetition

$\models$ (INTEND x a^* q) $\supset$ (INTEND x $a;(a)^*$ q)

Concurrent Actions

$\models$ (INTEND x $a_1||a_2||\ldots||a_n$ q) $\supset$
(INTEND x a_1 (INTEND x $a_1||a_2||\ldots||a_n$) q)) $\wedge$
(INTEND x a_2 (INTEND x $a_1||a_2||\ldots||a_n$ q)) $\wedge$
$\vdots$
(INTEND x a_n (INTEND x $a_1||a_2||\ldots||a_n$ q)) $\wedge$
[$\forall$k, 1$\leqslant$ k $\leqslant$ n (HAPPENS a_k) $\supset$ (HAPPENS a_1) $\wedge$...$\wedge$ (HAPPENS a_n))]

Non-Deterministic OR

$\models$ (INTEND x $a_1|a_2|\ldots|a_n$ q) $\supset$
(INTEND x a_1 (INTEND x $a_1|a_2|\ldots|a_n$ q)) $\vee$
(INTEND x a_2 (INTEND x $a_1|a_2|\ldots|a_n$ q)) $\vee$
$\vdots$
(INTEND x a_n (INTEND x $a_1|a_2|\ldots|a_n$ q))

therefore, the above results apply to PGOAL as well. For sake of completeness, the results

on commitment for doing an action expression are summarized in Table 2.2. The above

Table 2.2: Theorems on Commitment to do Action Expressions

Action Sequences

$\models$ (PGOAL x (DONE a_1;a_2;...;a_n) q) $\wedge$ $\neg$(BEL x (EARLIER (DONE a_1)))
$\supset$ (PGOAL x (DONE a_1 (PGOAL x a_1;a_2;...;a_n) q))

$\models$ (PGOAL x (DONE a_1;a_2;...;a_n) q) $\wedge$ (BEL x (DONE a_1;a_2;...;a_i))
$\supset$ (PGOAL x (DONE a_{i+1};...;a_n) (PGOAL x (DONE a_1;a_2;...;a_n) q))

Repetition

$\models$ (PGOAL x (DONE a*) q) $\supset$ (PGOAL x (DONE a;(a)*) q)

Concurrent Actions

$\models$ (PGOAL x (DONE $a_1||a_2||...||a_n$) q) $\supset$
(PGOAL x (DONE a_1) (PGOAL x (DONE $a_1||a_2||...||a_n$) q)) $\wedge$
(PGOAL x (DONE a_2) (PGOAL x (DONE $a_1||a_2||...||a_n$) q)) $\wedge$
$\vdots$
(PGOAL x (DONE a_n) (PGOAL x (DONE $a_1||a_2||...||a_n$) q)) $\wedge$
[$\forall$k, 1$\leqslant$ k $\leqslant$ n (HAPPENS a_k) $\supset$ (HAPPENS a_1) $\wedge$...$\wedge$ (HAPPENS a_n))]

Non-Deterministic OR

$\models$ (PGOAL x (DONE $a_1|a_2|...|a_n$) q) $\supset$
(PGOAL x (DONE a_1) (PGOAL x (DONE $a_1|a_2|...|a_n$) q)) $\vee$
(PGOAL x (DONE a_2) (PGOAL x (DONE $a_1|a_2|...|a_n$) q)) $\vee$
$\vdots$
(PGOAL x (DONE a_n) (PGOAL x (DONE $a_1|a_2|...|a_n$) q))

analysis of individual commitment and intention has been extended to multiple agents. An agent team is characterized as having joint commitment and joint intention as discussed next.

2.3.2 Joint Commitment and Joint Intention

The joint commitment between two agents is expressed formally by a joint persistent goal (JPG), and the one-sided commitment of an agent towards another agent is represented by a persistent weak achievement goal (PWAG). These terms are defined in terms of

mutual belief (MB), mutual goal (MG), and weak mutual goal (WMG). Mutual belief is expressed using unilateral mutual belief (BMB) that is treated as a semantic primitive [66]. A unilateral mutual belief (BMB x y p) says that agent x believes that there is mutual belief between itself and agent y about proposition p. The fixed-point definition of BMB is given in Definition 2.3 and a model-theoretic definition of BMB appears in Section 2.4.1. Mutual belief is discussed further in the next section. Mutual knowledge (MK) is defined as true mutual belief. Two agents have a mutual goal (MG) that p if they mutually believe that both agents have the goal that eventually p.

Definition 2.3. Unilateral Mutual Belief, Mutual Belief, Mutual Knowledge, and Mutual Goal

a. (BMB x y p) $\triangleq$ (BEL x $p \wedge$(BMB y x p))

b. (MB x y p) $\triangleq$ (BMB x y p) $\wedge$ (BMB y x p)

c. (MK x y p) $\triangleq$ $p \wedge$ (MB x y p)

d. (MG x y p) $\triangleq$ (MB x y (GOAL x $\Diamond p$)$\wedge$(GOAL y $\Diamond p$))

A weak mutual goal (WMG) is a mutual belief that each agent has a weak achievement goal (WAG) towards the other for achieving p relative to q.

Definition 2.4. Weak Mutual Goal

(WMG x y p q) $\triangleq$ (MB x y (WAG x y p q) $\wedge$ (WAG y x p q))

An agent x has a WAG towards another agent y when the following holds: if agent x believes that p is not currently true then it will have a goal to achieve p, and if it believes p to be either true, or to be impossible, or if it believes the relativizing condition q to be false, then it will have a goal to bring about the corresponding mutual belief with agent y.

Definition 2.5. Weak Achievement Goal

$$\begin{aligned}(\text{WAG x y } p\ q) \triangleq\ & [\neg(\text{BEL x } p) \wedge (\text{GOAL x } \Diamond p)] \vee \\ & [(\text{BEL x } p) \wedge (\text{GOAL x } \Diamond(\text{MB x y } p))] \vee \\ & [(\text{BEL x } \Box\neg p) \wedge (\text{GOAL x } \Diamond(\text{MB x y } \Box\neg p))] \vee \\ & [(\text{BEL x } \neg q) \wedge (\text{GOAL x } \Diamond(\text{MB x y } \neg q))]\end{aligned}$$

The joint commitment between two agents to achieve p is formally expressed as a joint persistent goal between those agents.

Definition 2.6. Joint Persistent Goal

$$(\text{JPG x y } p\ q) \triangleq (\text{MB x y } \neg p) \wedge (\text{MG x y } p) \wedge$$
$$(\text{UNTIL } [(\text{MB x y } p) \vee (\text{MB x y } \Box\neg p) \vee (\text{MB x y } \neg q)](\text{WMG x y } p\ q))$$

Two agents x and y have a joint persistent goal that p with respect to q when precisely the following conditions hold: there is a mutual belief that p is not currently true, it is a mutual goal to bring about p, and p will remain a weak mutual goal at least until there is a mutual belief that p is either true, or will never be true, or the relativizing condition q is no longer true. The agents can be jointly committed to doing an action (or actions) a because p can be of the form (DONE a). Note, that this applies even if the entire action (or actions) a is to be done by just one agent, say, by agent y as in (DONE y a). From [70], a joint commitment between two agents implies individual commitment by each agent, assuming the relativizing condition to be the true, that is,

Proposition 2.1. $\models (\text{JPG x y } p) \supset (\text{PGOAL x } p) \wedge (\text{PGOAL y } p)$

Therefore, a JPG between x and y for (DONE y a) also commits agent x to agent y's doing the action a. Agent x can then act on its individual commitment (PGOAL x (DONE y a)) by being co-operative, helpful, or in any other suitable manner. For example, if the joint commitment is for agent y to move a chair out of the room, then agent x can act proactively to help agent y and open the door if the door is closed. This behavior is in addition to the mutual belief establishment by both parties in the event of private belief regarding achievement, impossibility, or irrelevance of the jointly committed goal. Two agents having a JPG to do an action are said to form a *team* to do that action [70]. Joint commitment defined above is not a social commitment – in a social commitment [105], an agent (called the debtor) may have a one-way commitment towards another agent (called the creditor) for doing an action but the creditor need not have the same commitment towards the debtor. If two agents are jointly committed to one of them doing a certain action, the commitment binds both agents towards each other for doing that action and it prescribes similar response from both agents in the event that one agent privately comes

to believe that the jointly committed action has already been achieved or is impossible or irrelevant.

Joint Intention (JI) between two agents is defined in terms of the joint persistent goal and the definition of JI makes the jointly committed action explicit.

Definition 2.7. Joint Intention

$$(\text{JI x y } a\ q) \triangleq$$
$$(\text{JPG x y (DONE x y [UNTIL (DONE x y } a)(\text{MB x y (DOING x y } a))]?; a)\ q)$$

Two[3] agents x and y jointly intend to do an action (or actions) a if they have a joint commitment to doing the action a mutually believing throughout the action execution that they are jointly doing that action as a team [70]. Joint intention requires a starting mutual belief that the agents are about to start the jointly intended action. JI is defined in terms of JPG and therefore, a JI has the same property as a JPG regarding the establishment of appropriate mutual beliefs in the event of private beliefs about achievement, impossibility, or irrelevance of the jointly intended action. Also, as with JPG, the jointly intended action does not necessarily have to be a joint action[4], that is, it can be an action requiring only one agent to act, in which case the non-acting agent has commitments towards the other agent's actions. For example, in the analysis of protocols in Chapter 8, we treat a conversation protocol as a joint action expression represented by the action a in the definition of joint intention and we assume that the participants of a conversation protocol jointly intend to execute that conversation protocol.

The consequences of jointly intending an action expression has been analyzed in detail in [70, 29, 28]. It has been shown that if a team jointly intends to do an action and if one member of the team believes that he is the only agent of that action then that member privately intends to do the action. This is because joint commitment entails individual commitment and mutual belief entails individual belief. Furthermore, the agent must believe that he is the only agent of that action because an agent is not allowed to intend

[3]Joint intention is defined here for two agents. For multiple agents, a generalized version of joint intention that uses a schema variable for teams in place of individual agents in the above definition is defined in Chapter 4.

[4]Joint action refers to an action that requires both agents to act, for instance, an action expression involving both agents is referred to as a joint action expression.

to perform other agent's actions (although they can be committed to actions of other agents as discussed earlier).

Similarly, if a team jointly intends to do a concurrent action that consists of the team members concurrently doing individual sub-actions, then the individual members will privately intend to do their parts relative to the overall joint intention. As such, these individual intentions persist as long as the joint intention persists. The definition of joint intention requires that agents jointly intending an action mutually believe throughout that they are doing the jointly intended action. Therefore, it follows that even when the agents are performing their respective individual actions as part of the jointly intended action, they also believe throughout that they are performing the group action together.

Similar to that for individual intention, the joint execution of more complex action expressions such as action sequences and repetitive actions offers individual agents lots of flexibility in executing the action expression. For instance, agents jointly intending an action sequence are not required to mutually believe when each action in the sequence has been done. As in the case of individual intention, this flexibility can be restricted using various execution strategies.

Joint Stepwise Execution Strategy: A joint stepwise execution strategy (also called *lockstep*) requires attainment of mutual belief after each step that the step had been accomplished and that the agents were embarking on the remainder. However, we do not require a team to always execute complex actions in lockstep as there are many types of joint actions where such a team overhead is undesirable. In fact, the formalism allows for individuals to contribute privately to a sequence when that is compatible with the performance of the overall activity. If a team jointly intends to do a sequential action, then the agent of any part will intend to do that part relative to the larger intention provided that she will always know when the antecedent part is over, when she is doing her share, and when she has done it. Therefore, if a team jointly intends to do a sequence of actions in a joint stepwise fashion, the agents of any of the steps will jointly intend to do the step relative to the larger intention.

Joint commitments and joint intentions are typically established using communication wherein each agent communicates to the other agent that it has a one-way commitment

towards the other agent. This one-way commitment of one agent towards another agent is called persistent weak achievement goal and is discussed next.

2.3.3 Social Commitment (Persistent Weak Achievement Goal)

A persistent weak achievement goal or PWAG represents the one-way commitment of one agent towards another. As such, PWAG is the building block for joint commitment in multi-agent communication – two agents will be jointly committed to achieving a goal (i.e., they will have a JPG) if they mutually believe that both the agents have an appropriate PWAG, with respect to each other to achieve that goal (Theorem 2.1).

Definition 2.8. Persistent Weak Achievement Goal

$$\begin{aligned}(\text{PWAG x y } p\ q) \triangleq\ & [\neg(\text{BEL x } p) \wedge (\text{PGOAL x } p\ q)] \vee \\ & [(\text{BEL x } p) \wedge (\text{PGOAL x } (\text{MB x y } p)\ q)] \vee \\ & [(\text{BEL x } \Box\neg p) \wedge (\text{PGOAL x } (\text{MB x y } \Box\neg p)\ q)] \vee \\ & [(\text{BEL x } \neg q) \wedge (\text{PGOAL x } (\text{MB x y } \neg q))]\end{aligned}$$

This definition, modified from Smith & Cohen [106], states that an agent x has a PWAG towards another agent y when the following holds: if agent x believes that p is not currently true then it will have a persistent goal to achieve p, and if it believes p to be either true, or to be impossible, or if it believes the relativizing condition q to be false, then it will have a persistent goal to bring about the corresponding mutual belief with agent y. Note that the PGOAL in the first three disjuncts are relative to q and therefore, establishing a mutual belief that $\neg q$ allows the PWAG to be dropped. A PWAG is more than a commitment to performing an action and more than the commitment to perform an action in concert with another agent — not only is the agent committed to achieving p but, once p is achieved or dropped, the agent will acquire a persistent goal to achieve mutual belief that the goal has been achieved (or is impossible or no longer relevant). Two agents having inter-locking PWAGs towards each other form a joint commitment with each other. We state this in the following theorem and use the result in the rest of this dissertation.

Theorem 2.1. *Mutual belief in each other's PWAG towards the other to achieve a goal p is sufficient to establish a joint commitment to achieve p provided that* (1) *there is mutual*

belief that p has not already been achieved, and (2) *the PWAGs are interlocking, that is, one PWAG is relative to the other. Formally,*

$$\models [(\text{MB x y (PWAG x y } p\ q)) \wedge (\text{MB x y (PWAG y x } p\ r{\wedge}q)) \wedge (\text{MB x y } \neg p)]$$
$$\supset (\text{JPG x y } p\ r{\wedge}q)$$
$$\text{where } r = (\text{PWAG x y } p\ q)$$

Proof. We need to show that all three conjuncts in the definition of (JPG x y p $r{\wedge}q$) follow from the premise. Using the definition of JPG (Definition 2.6), we see that (JPG x y p $r{\wedge}q$) requires the following conjuncts:

- (MB x y $\neg p$)
- (MG x y p)
- (UNTIL [(MB x y p) $\vee$ (MB x y $\Box\neg p$) $\vee$ (MB x y $\neg(r{\wedge}q)$)] (WMG x y p $r{\wedge}q$))

We need to show that all these three conjuncts are satisfied by the premise of this theorem.

(1) (MB x y $\neg p$) is the assumed initial condition. Hence this condition is trivially satisfied.

(2) In order to show that (MG x y p) is satisfied, using definition of MG (Definition 2.3d), we need to show that the following conjunction is satisfied:

$$(\text{MB x y (GOAL x } \Diamond p)) \wedge (\text{MB x y (GOAL y } \Diamond p))$$

First note the following property of mutual belief:

(2a) $\models (\text{MB x y } \neg p) \supset (\text{MB x y (BEL x } \neg p)) \wedge (\text{MB x y (BEL y } \neg p))$

Now consider the first and last conjunct in the premise of this theorem:

(2b) (MB x y (PWAG x y p q)) $\wedge$ (MB x y $\neg p$)

$\supset$ (MB x y (PWAG x y p q) $\wedge$ (BEL x $\neg p$)) [By (2a) & distrib. of MB over conj.]

$\supset$ (MB x y (PGOAL x p)) [By Defn. 2.8 of PWAG]

$\supset$ (MB x y (GOAL x $\Diamond p$)) [By Defn. 2.1 of PGOAL]

Similarly, from the second and the last conjunct in the premise,

(2c) $(\text{MB x y (PWAG y x } p\ r{\wedge}q)) \wedge (\text{MB x y } \neg p) \supset (\text{MB x y (GOAL y } \Diamond p))$

Therefore, from (2b), (2c), and (Definition 2.3d) of mutual goal, (MG x y p) is satisfied.

(3) In order to establish the last conjunct (the UNTIL clause), we first show that (WMG x y $p\ r{\wedge}q$) is satisfied by the given premise. Using definition of WMG (Definition 2.4), we want to show that the following conjunction is satisfied:

(3a) $(\text{MB x y (WAG x y } p\ r{\wedge}q)) \wedge (\text{MB x y (WAG y x } p\ r{\wedge}q))$

An agent having a persistent goal that p has a goal that eventually p, that is,

(3b) $\models (\text{PGOAL x } p) \supset (\text{GOAL x } \Diamond p)$ [From Definition 2.1 of PGOAL]

Therefore, from the definition of WMG (Definition 2.4) and the definition of PWAG (Definition 2.8), we see that

(3c)
$\models (\text{PWAG x y } p\ q) \supset (\text{WAG x y } p\ q)$ [Using (3b) & definitions of PWAG and WMG]

We immediately see that the second conjunct in (3a) follows from the second conjunct in the premise

(3d) $(\text{MB x y (PWAG y x } p\ r{\wedge}q)) \supset (\text{MB x y (WAG y x } p\ r{\wedge}q))$ [Using (3c)]

Similarly, in order to show that the first conjunct in (3a) follows from the first conjunct in the premise, we need to establish that the following holds:

(3e) $(\text{MB x y (PWAG x y } p\ q)) \supset (\text{MB x y (PWAG x y } p\ r{\wedge}q))$

From the definition of PWAG (Definition 2.8), we see that the first three disjuncts are common to the PWAG in the LHS and the RHS of (3e). The only difference is in the last disjunct involving the relativizing condition. The last disjunct of the PWAG in the LHS of (3e) is

$(\text{BEL x } \neg q) \wedge (\text{PGOAL x (MB x y } \neg q))$

The last disjunct of the PWAG in the RHS of (3e) is

$(\text{BEL x } \neg(r{\wedge}q)) \wedge (\text{PGOAL x (MB x y } \neg(r{\wedge}q)))$

Therefore, (3e) can be established if we can show that

(3f) $(\text{BEL x } \neg q) \wedge (\text{PGOAL x (MB x y } \neg q))$
$\supset (\text{BEL x } \neg(r{\wedge}q)) \wedge (\text{PGOAL x (MB x y } \neg(r{\wedge}q)))$

We can conclude (3f) by applying OR-introduction[5] to both the conjuncts in LHS of (3f). Therefore, we can conclude (3e), that is,

(MB x y (PWAG x y p q)) $\supset$ (MB x y (PWAG x y p $r \wedge q$))

From the premise of this theorem, we have

(MB x y (PWAG x y p q)) $\wedge$ (MB x y (PWAG y x p $r \wedge q$))

$\supset$ (MB x y (PWAG x y p $r \wedge q$)) $\wedge$ (MB x y (PWAG y x p $r \wedge q$)) [Using (3e)]

$\supset$ (MB x y (WAG x y p $r \wedge q$)) $\wedge$ (MB x y (WAG y x p $r \wedge q$)) [Using (3d)]

= (WMG x y p $r \wedge q$)) [Using Definition 2.4 of WMG]

Now this WMG persists at least until the mutual belief in PWAGs that it is derived from persist. Therefore, it persists at least until

(MB x y (PWAG x y p q)) and (MB x y (PWAG y x p $r \wedge q$)) persist.

Using the implication (3d) established above, WMG persists at least until

(MB x y (PWAG x y p $r \wedge q$)) and (MB x y (PWAG y x p $r \wedge q$)) persists.

That is, WMG persists at least until

(MB x y p) $\vee$ (MB x y $\Box \neg p$) $\vee$ (MB x y $\neg(r \wedge q)$) persists.

[Using Definition 2.8 of PWAG]

Therefore, we can conclude that

(MB x y (PWAG x y p q)) $\wedge$ (MB x y (PWAG y x p $r \wedge q$)) $\wedge$ (MB x y $\neg p$)

$\supset$ (UNTIL [(MB x y p) $\vee$ (MB x y $\Box \neg p$) $\vee$ (MB x y $\neg(r \wedge q)$)]

(WMG x y p $r \wedge q$)))

This establishes the final conjunct (the UNTIL clause) in the definition of JPG to be proved. □

A consequence of this theorem is that it is possible to establish joint commitment between two agents by using communicative acts to establish mutual belief in each other's PWAG towards the other. We will see later in this dissertation that it is possible to establish mutual belief by default about one's PWAG using a single communicative act (i.e., without confirmation from the other agent). However, in order to establish that result, we need to first show that agents are competent with respect to their PWAG as stated formally in the following lemma.

[5]Without loss of generality, we will assume throughout this paper that all agents have access to the same inference rules as we do. For non-reasoning agents, this would mean that the agent designer has access to the same inference rules as we do.

Lemma 2.1. *If an agent believes that it has a PWAG towards another agent, then it does have that PWAG towards that agent. Formally,*

$$\models (\text{BEL x } (\text{PWAG x y } p\ q))\ \supset\ (\text{PWAG x y } p\ q)$$

Proof. This lemma follows from the definition of PWAG and the assumption that agents are competent with respect to their individual commitment (Assumption 2.1). First, recall that an agent is competent with respect to its individual commitment, that is,

(1) $(\text{BEL x } (\text{PGOAL x } p)) \supset (\text{PGOAL x } p)$ [Assumption 2.1]

We want to show that the LHS and RHS lead to the same mental attitudes whenever any disjunct in the definition of PWAG (Definition 2.8) is true. We need to consider four cases – one for each disjunct in the definition of PWAG.

Case 1: $\neg(\text{BEL x } p)$

In this case,

$(\text{BEL x } (\text{PWAG x y } p\ q)) \wedge \neg(\text{BEL x } p)$

$\supset (\text{BEL x } (\text{PWAG x y } p\ q)) \wedge (\text{BEL x } \neg(\text{BEL x } p))$ [Negative introspection]

$\supset (\text{BEL x } (\text{PWAG x y } p\ q) \wedge \neg(\text{BEL x } p))$ [Belief distributes over conjunction]

$\supset (\text{BEL x } (\text{PGOAL x } p\ q))$ [From definition of PWAG]

$\supset (\text{PGOAL x } p\ q)$ [From (1)]

$\therefore (\text{PWAG x y } p\ q) \wedge \neg(\text{BEL x } p) \supset (\text{PGOAL x } p\ q)$ [Using definition of PWAG]

Case 2: $(\text{BEL x } p)$

In this case,

$(\text{BEL x } (\text{PWAG x y } p\ q)) \wedge (\text{BEL x } p)$

$\supset (\text{BEL x } (\text{PWAG x y } p\ q)) \wedge (\text{BEL x } (\text{BEL x } p))$ [Positive introspection]

$\supset (\text{BEL x } (\text{PWAG x y } p\ q) \wedge (\text{BEL x } p))$ [Belief distributes over conjunction]

$\supset (\text{BEL x } (\text{PGOAL x } (\text{MB x y } p)\ q))$ [From definition of PWAG]

$\supset (\text{PGOAL x } (\text{MB x y } p)\ q)$ [From (1)]

$\therefore (\text{PWAG x y } p\ q) \wedge (\text{BEL x } p) \supset (\text{PGOAL x } (\text{MB x y } p)\ q)$ [Using definition of PWAG]

Case 3: $(\text{BEL x } \Box\neg p)$

In this case,

(BEL x (PWAG x y p q)) $\wedge$ (BEL x $\Box\neg p$)

$\supset$ (BEL x (PWAG x y p q)) $\wedge$ (BEL x (BEL x $\Box\neg p$)) [Positive introspection]

$\supset$ (BEL x (PWAG x y p q) $\wedge$ (BEL x $\Box\neg p$)) [Belief distributes over conjunction]

$\supset$ (BEL x (PGOAL x (MB x y $\Box\neg p$) q)) [From definition of PWAG]

$\supset$ (PGOAL x (MB x y $\Box\neg p$) q) [From (1)]

$\therefore$ (PWAG x y p q) $\wedge$ (BEL x $\Box\neg p$) $\supset$ (PGOAL x (MB x y $\Box\neg p$) q) [Using definition of PWAG]

Case 4: (BEL x $\neg q$)

In this case,

(BEL x (PWAG x y p q)) $\wedge$ (BEL x $\neg q$)

$\supset$ (BEL x (PWAG x y p q)) $\wedge$ (BEL x (BEL x $\neg q$)) [Positive introspection]

$\supset$ (BEL x (PWAG x y p q) $\wedge$ (BEL x $\neg q$)) [Belief distributes over conjunction]

$\supset$ (BEL x (PGOAL x (MB x y $\neg q$))) [From definition of PWAG]

$\supset$ (PGOAL x (MB x y $\neg q$)) [$From(1)$]

$\therefore$ (PWAG x y p q) $\wedge$ (BEL x $\neg q$) $\supset$ (PGOAL x (MB x y $\neg q$)) [Using definition of PWAG]

These four cases exhaustively cover all situations relevant for a PWAG. We see that in each of these cases,

(BEL x (PWAG x y p q)) results in the same mental attitude as (PWAG x y p q).

Hence, we can conclude that

(BEL x (PWAG x y p q)) $\supset$ (PWAG x y p q) $\Box$

PWAG is a central concept in the Joint Intention theory and is used in the definition of the various communicative acts, in particular, the request communicative act. A PWAG defines a commitment of one agent towards another and therefore it represents a social commitment, provided that it is made public[6]. As such, a PWAG reflects an agent's social

[6]Making a PWAG public means making it observable by others. In the simplest case, it means establishing a mutual belief between the agent who has the PWAG and the agent towards whom the PWAG is directed. In general, however, it requires group communication (Chapter 5) where the senders, the listeners, and the over-hearers can be groups of agents.

commitment or obligation towards other parties, even though it is a commitment defined using mental attitudes. The PWAG of an agent towards other agents can be inferred by listening to its observable communication (i.e., if the communication is not private and encrypted). Furthermore, one can verify to a certain extent whether or not an agent acts according to its PWAG by setting up test cases where the test agents establish mutual beliefs to satisfy the different conditions of the PWAG and then observing the agent's communicative as well as other external actions. Commitments and intentions can be made public by establishing appropriate mutual beliefs. Mutual belief can be established by either co-presence[7] or by explicit communication. However, explicit communication by message passing is the most prevalent way of establishing mutual belief in multi-agent systems. Next, we take a detailed look at the concept of mutual belief and how it can be established by message passing.

2.4 MUTUAL BELIEF

Mutual belief is defined in terms of unilateral mutual belief. Two agents x and y are said to mutually believe that p if both agents believe that they mutually believe that p. That is,

$$(\text{MB x y } p) \quad \triangleq \quad (\text{BMB x y } p) \wedge (\text{BMB y x } p)$$

This definition is symmetric so we will just discuss (BMB x y p) in what follows. We used the following fixed-point definition of BMB in this chapter (Definition 2.3a),

$$(\text{BMB x y } p) \quad \triangleq \quad (\text{BEL x } p \wedge (\text{BMB y x } p))$$

This fixed point definition encodes the following well-known property of mutual belief: An agent x believes that it is mutually believed between itself and another agent y that p *iff* agent x believes p, and believes that agent y believes p, and believes that y believes that x believes p, and so on, ad infinitum. By expanding the fixed-point definition using belief distribution over conjunction, one can readily verify the following proposition.

[7] One may treat co-presence as a form of communication without explicit message passing.

Proposition 2.2. Unilateral mutual belief as an infinite conjunction

$$(\text{BMB x y } p) \supset (\text{BEL x } p) \wedge (\text{BEL x (BEL y } p)) \wedge (\text{BEL x (BEL y (BEL x } p))) \wedge \ldots \text{ and so on}$$

We first give a model theoretic definition of BMB and show that the fixed-point property of BMB follows from this definition. The model theoretic definition allows finite models of mutual belief that can be represented and reasoned about in software systems.

2.4.1 Model-Theoretic Definition of BMB

We assume a model structure $\mathbb{M}$ from Section 2.2 that includes an accessibility relation $\mathcal{B}_a$ for every agent a. We use the usual possible worlds representation where $\omega_1 \mathcal{B}_a \omega_2$ means that world ω_2 is belief accessible by agent a from world ω_1 As mentioned earlier, the belief accessibility relation $\mathcal{B}_a$ is constrained to be Euclidean, transitive, and serial in our model from [25]. In this model, an agent x has unilateral mutual belief (BMB) with agent y about proposition p if and only if $\forall \omega_1$, $\omega_2 \ldots \omega_n$ such that $\omega_1 \mathcal{B}_x \omega_2$, $\omega_2 \mathcal{B}_y \omega_3$, $\omega_3 \mathcal{B}_x \omega_4$, $\omega_4 \mathcal{B}_y \omega_5$,$\ldots \omega_{n-1} \mathcal{B}_a \omega_n$ (where $\mathcal{B}_a$ is $\mathcal{B}_x$ if n is even or $\mathcal{B}_y$ if n is odd), p is valid in the model $\mathbb{M}$ in world ω_n. More formally,

$$\mathbb{M}, \omega \models (\text{BMB x y } p) \triangleq \forall n \, \forall(\omega, \omega') \, \varepsilon \, \mathcal{B}[x, y, n] \;\; \mathbb{M}, \omega' \models p$$

where, $\mathcal{B}[x, y, n]$ is defined inductively by

$$\mathcal{B}[x, y, 1] = \mathcal{B}_x$$

$$\mathcal{B}[x, y, n+1] = \mathcal{B}[y, x, n] \circ \mathcal{B}_x$$

This semantics of BMB is similar to the semantics of common knowledge given in [48]. To show that the fixed-point property of BMB follows from this model-theoretic definition, consider a world ω_1 that is belief accessible from ω by agent x (i.e., $\omega \mathrm{B}_x \omega_1$). From the definition of the above model, we have,

$\forall$n $\forall(\omega,\omega') \, \varepsilon \, \mathcal{B}[x, y, n]$ $\mathbb{M},\omega' \models p$

Now, consider all worlds ω' that are belief accessible by agent y from world ω_1, that is, $\omega_1 \mathcal{B}_y \omega'$.

Let m = n+1

We can conclude that,

$$\forall m\ \forall(\omega_1, \omega')\ \varepsilon\ \mathcal{B}[x, y, m]\ \mathbb{M}, \omega' \models p$$

where,

(1) $\mathcal{B}[y, x, 1] = \mathcal{B}_y$

(2) $\mathcal{B}[y, x, m+1] = \mathcal{B}[x, y, n] \circ \mathcal{B}_y$

This is nothing but the model definition of (BMB y x p). Since this is true in any arbitrary ω_1 that is belief accessible by x from ω, therefore, (BMB y x p) is true in all worlds that are belief accessible from ω by agent x. Hence, from the model, we can conclude that (BEL x (BMB y x p)).

Moreover, from the above definition we see that the following holds,

$$\forall(\omega, \omega')\ \varepsilon\ \mathcal{B}[x, y, 1]\ \mathbb{M}, \omega' \models p$$

That is, we can conclude (BEL x p) from this model. Therefore, from the above model, we can conclude that

$$\text{(BEL x (BMB y x } p\text{))}\ \wedge\ \text{(BEL x } p\text{)}$$

This is the fixed-point definition of (BMB x y p).

A model for (MB x y p) can be obtained by combining the models for (BMB x y p) and (BMB y x p). Note that the above model for (BMB x y p) is an infinite model. However, it is possible to give finite models for (BMB x y p) by suitably modifying the above infinite model such that the fixed-point property is retained. One such model can be obtained by constraining the belief accessibility relation such that

(1) It is reflexive, that is,

$$\forall\omega, \omega_1\ \omega\mathcal{B}_x\omega_1 \supset \omega_1\mathcal{B}_x\omega_1$$

(2) It "loops-back" in the following manner:

$$\forall\ \omega, \omega_1, \omega_2, \omega_3\ \ \omega\mathcal{B}_x\omega_1\ \wedge\ \omega_1\mathcal{B}_y\omega_2\ \wedge\ \omega_2\mathcal{B}_x\omega_3\ \supset\ \omega_3\mathcal{B}_y\omega_2$$

Any number of such models can be obtained depending on how many levels of private beliefs agent x needs to have. If ω is the real world, then there is only one level of private belief. If ω is directly belief accessible from the real world ω_0, that is, $\omega_0\mathcal{B}_x\omega$

then agent x has two levels of private belief, and so on. Such finite models of BMB can be readily translated into finite data-structures that can be used to represent and reason about mutual belief. One such circular data-structure due to Cohen [22] demonstrates the feasibility of representing and reasoning about mutual beliefs. The next step is to establish mutual belief between agents by explicit message passing as well as by other communication modalities such as by observation.

2.4.2 Establishing Mutual Belief by Communication

Mutual belief is sometimes thought of as an infinite conjunction of beliefs as in Proposition 2.2. This view of mutual belief has led some researchers [104, 105] to argue that mutual belief can never be established via message passing. However, the basic assumption of these researchers that each of the conjuncts in Proposition 2.2 has to be established one by one in order to establish mutual belief, is incorrect. For example, Cohen [22] argued that, rather than attain (BMB x y p) incrementally, the infinite number of levels can be assumed to hold at once, provided there is no (finite) level of belief embedding that p is false. In that work, a circular data structure of databases encoded the infinite repetition of alternating beliefs in Proposition 2.2. Brainov and Sandholm [9] have independently proposed a similar technique to represent and reason about infinite belief hierarchies using finite data-structures. As such, mutual belief can be represented using finite data structures, and there exist finite algorithms to reason about mutual belief using those data structures.

It has been shown in the literature that mutual belief can be established in several different ways by default [22, 83, 56, 76, 77], even if the communication channel is fallible. In this dissertation, we assume that mutual belief is established by default. Establishing mutual belief by default means that agents make certain defeasible assumptions if they have no information to the contrary. For example, in certain situations, an agent can send a message and assume that the message was delivered to the recipient. Mutual belief is, then, established quite easily based on such defeasible beliefs. These beliefs can be false but belief (as opposed to knowledge) is allowed to be incorrect. If, at a later time, the agent who has a defeasible belief discovers that its belief was incorrect, then it needs

to revise that belief. But until that happens, an agent can assume the defeasible belief and make inferences based on that belief. We will formally state the defeasible rules of communication that an agent may use to establish mutual belief by default using terms that we define next.

Definition 2.9. Sincere
An agent x is sincere towards another agent y with respect to a proposition p if, whenever x wants y to come to believe p, it wants y to come to know p. Formally,

$$\begin{aligned}(\text{SINCERE x y } p) \triangleq\ & \forall\, e\ (\text{GOAL x } (\text{HAPPENS } e; (\text{BEL y } p)?)) \\ & \supset (\text{GOAL x } (\text{HAPPENS } e; (\text{KNOW y } p)?))\end{aligned}$$

Therefore, a sincere agent only communicates information that it believes to be true at the time of communication. In defeasible reasoning about communication, an agent may believe something by default if it has no belief to the contrary. This concept is explicated precisely using unilateral mutual ignorance (BMI).

Definition 2.10. Unilateral Mutual Ignorance
An agent x believes that there is mutual ignorance[8] between itself and another agent y about a proposition p iff it does not believe that p, and does not believe that y believes p, and does not believe that y believes that x believes p, and so on, ad infinitum. Formally,

$$\begin{aligned}(\text{BMI x y } p) \triangleq\ & \neg(\text{BEL x } p) \ \wedge\ \neg(\text{BEL x } (\text{BEL y } p)) \\ & \wedge\ \neg(\text{BEL x } (\text{BEL y } (\text{BEL x } p))) \wedge \ldots \text{ and so on}\end{aligned}$$

Even though BMI is defined above as an infinite conjunction, it can be reasoned about and represented using finite data structures, and can be given a model theoretic definition using finite models (as in the case of BMB). We have shown in the previous section how to create finite models that satisfy BMB. Presence of a proposition p in the possible worlds at all levels in such a finite model implies (BMB x y p) and its absence (i.e., if it cannot be inferred) from the possible worlds at all levels in that model implies (BMI x y p). An agent x has no information to the contrary about a proposition p if, for every agent y that x has beliefs about, agent x believes that x and y are mutually ignorant that $\neg p$, that is,

[8] Our usage of the term "mutual ignorance" differs slightly from Johnson-Laird [54] where it refers to absence of mutual knowledge.

if $\forall$y (BMI x y $\neg p$) holds. In this chapter, we only consider communication between two agents x and y, and so, (BMI x y $\neg p$) is sufficient to conclude that agent x has no belief to the contrary about p.

Defeasible Rules about Communication

Here we introduce the notation and specify the defeasible rules of communication that are used to establish results about communicative acts later in this dissertation.

Definition 2.11. Defeasible Implication ($\Rightarrow$)
We use the double arrow notation ($\varphi \Rightarrow \psi$) to denote defeasible rules about communication. The satisfaction relation $\models (\varphi \Rightarrow \psi)$ then denotes that ψ can always be concluded from φ in our model using the defeasible rules that we state next.

We will see in the next section that every communicative act has an associated goal and an intention. In what follows, let α(x,y,e,ϕ,ψ) be a communicative act where x is the sender (speaker), y is the recipient (hearer), e is the event of performing that act, ϕ is the associated goal, and ψ is the associated intention of the act. We assume that the sender and the recipient mutually believe that both the agents use the following defeasible rules (Propositions 2.3-2.8).

The performer of a communicative act believes after sending the message that it is mutually believed between the sender and the recipient that the communicative act has been done successfully. A communicative act is successful if the hearer recognizes the speaker's goal and intention associated with performing that communicative act [100, 19]. In the present work, however, we do not formally distinguish between the performance and the success[9] of communicative acts, that is, the term (DONE a) where a is a communicative act, denotes not only that a has just been done but also that it has been successful. In other words, the speaker assumes perfect communication. We will model this as a defeasible assumption formally stated as,

(a) (DONE α(x,y,e, ϕ,ψ)) $\Rightarrow$ (BMB x y (DONE α(x,y,e, ϕ,ψ)))

[9]Separating these two concepts would require formally defining "success" and recognizing that there is possibly a time lag between the receipt of a message by a hearer and his actual processing of that message.

If the hearer y receives (and processes) the message, then it believes that the communicative act has been done, that is, (BEL y (DONE α(x,y,e, ϕ, ψ)) holds. From mutual belief that each agent uses these default rules, the recipient believes that the sender believes that it is mutually believed between the sender and the recipient that the communicative act has been done, that is, (BEL y (BMB x y (DONE α(x,y,e, ϕ, ψ)))) holds. Therefore, we have

(b) (DONE α(x,y,e, ϕ,ψ)) $\Rightarrow$ (BMB y x (DONE α(x,y,e, ϕ,ψ))) [From definition of BMB]

Hence, we can defeasibly conclude that, if a communicative act has been done (i.e., the message has been received), then mutual belief occurs by default that the communicative act has been done. Formally,

Proposition 2.3.

(DONE α(x,y,e, ϕ,ψ)) $\Rightarrow$ (MB x y (DONE α(x,y, e, ϕ,ψ))) [From (a) and (b)]

We assume that the speaker of a communicative act does not change its mind, immediately after performing the communicative act, about any proposition whose truth-value is not affected by performance of that communicative act. For example, if a sender informs that p then p is the proposition of interest whose truth value is assumed to be unaffected by the act of informing. In order to state this assumption precisely, we need to address the Frame problem [92]. However, without worrying about whether or not the agent changed its mind while the communicative act was being performed, we state the following useful consequence of this assumption: If the recipient believes it is mutually believed that the sender believed p just before performing the communicative act and if the truth value of the sender's belief is not affected by performance of that communicative act, then after the communicative act is performed, it believes that it is mutually believed that the sender believes p. Formally,

Proposition 2.4.

$$\begin{aligned} \models \forall e\ (\text{DONE } \alpha(\text{x,y,}e, \phi, \psi)) \quad &\wedge \quad (\text{BMB y x (BEFORE } e \text{ (BEL x } p))) \\ &\wedge \quad \neg(\text{DONE (BEL x } p)?; e; \neg(\text{BEL x } p)?) \\ &\Rightarrow \quad (\text{BMB y x (BEL x } p)) \end{aligned}$$

The sender of a communicative act is sincere towards the recipient of that act about any mental attitude θ of the sender (for example, θ can be a PWAG of the sender). If the communicative intention ψ of a sender is to bring about mutual belief that before performing the communicative act, it had the goal that after the act is done, the recipient will believe θ, then the sender is assumed to be sincere towards the recipient with respect to θ. From the definitions of communicative acts in the next section, we will see that θ is either a belief or a PWAG of the sender.

Proposition 2.5. (DONE α(x,y, e, ϕ,ψ)) $\Rightarrow$ (SINCERE x y θ)
where, ψ = (BMB y x [BEFORE e (GOAL x (AFTER e (BEL y θ)))])

If the communicative intention of the performer of a communicative act is that the recipient believe that it is mutually believed that before performing the communicative act, the sender had the goal that after the act is performed, the recipient believes θ (where θ is a mental attitude of the sender as above), then after the act is performed, the recipient believes that it is mutually believed that θ, provided that (1) the recipient has no belief to the contrary about θ, and (2) the sender is sincere towards the recipient with respect to θ. The recipient has no belief to the contrary about θ if it believes that the sender and the recipient are mutually ignorant about $\neg\theta$. Formally,

Proposition 2.6. (DONE α(x,y,e, ϕ,ψ)) $\wedge$ (BMI y x $\neg\theta$) $\Rightarrow$ (BMB y x θ)
where, ψ = (BMB y x [BEFORE e (GOAL x (AFTER e (BEL y θ)))])

The sincerity condition is implicit in Proposition 2.6 because the premise of Proposition 2.5 is satisfied by the premise of this proposition. We assumed that the sender and the recipient mutually believe that both agents use these defeasible rules. Therefore, if the sender believes that the recipient does not have any belief to the contrary about the sender's mental attitude θ, then the sender will believe that the recipient believes that it is mutually believed that θ.

Proposition 2.7.
(DONE α(x,y, e,ϕ,ψ)) $\wedge$ (BEL x (BMI y x $\neg\theta$)) $\Rightarrow$ (BEL x (BMB y x θ))
where, ψ = (BMB y x [BEFORE e (GOAL x (AFTER e (BEL y θ)))])

A sender of a communicative act can assume that the recipient does not have any belief to the contrary about the sender's mental attitude θ that has just been communicated to the recipient by the communicative act. From the definitions of communicative acts in the next section, we will see that θ is always either the sender's belief or his PWAG towards the recipient. The equality (=) in the next proposition refers to propositional equality.

Proposition 2.8. (DONE α (x,y,e, ϕ,ψ)) $\wedge$ [(θ = (BEL x γ)) $\vee$ (θ = (PWAG x λ q))]
$\Rightarrow$ (BEL x (BMI y x $\neg\theta$))
where, ψ = (BMB y x [BEFORE e (GOAL x (AFTER e (BEL y θ)))]), and
γ, λ, and q denote some propositions

We will use these defeasible rules for the subsequent chapters. One consequence of these rules is that it may take only two messages to establish mutual belief in each other's PWAG, and thus to create a joint commitment due to the interlocking PWAG (Theorem 2.1). For all the theorems about communication that we establish in this dissertation, we will assume that the recipient does not have any belief to the contrary and that the truth-value of the proposition being communicated is not altered by the very act of communication.

Assumption 2.2. *We will assume that the following conditions hold for each communicative act $\alpha(x, y, e, \phi, \psi)$, where*
ψ = (BMB y x [BEFORE e (GOAL x (AFTER e (BEL y θ)))]]), *and*
the mental attitude θ of the sender x is of the form (PWAG x y p q) *or* (BEL x p)

(a) The recipient has no belief to the contrary about the proposition θ, that is,

$$\text{(BMI y x } \neg\theta\text{)}$$

(b) The communicative act does not change the truth value of the proposition θ, that is,

$$\neg(\text{HAPPENS } \theta?; e; \neg\theta?), \text{ and}$$
$$\neg(\text{DONE } \theta?; e; \neg\theta?)$$

Assumptions 2.2a and 2.2b are not defeasible rules as in Propositions (2.3-2.8), rather they are conditions that we assume throughout this dissertation in order to avoid re-stating them in the premise of every theorem. Mutual belief about PWAGs as well as about any

other attitude or proposition is established using an appropriate communicative act with appropriate content. Next, we introduce communicative acts and define those acts that are used in the Request protocol in the next chapter.

2.5 COMMUNICATIVE ACTS AS ATTEMPTS

In the philosophy of language, it is argued that the illocutionary effect of a speech act consists of the hearer's recognition of the speaker's communicative intention [100]. A communicative act succeeds when the hearer successfully recognizes the speaker's intention and it is satisfied when the hearer successfully acts on the speaker's intention. Communicative acts must be characterized as attempts because there is a possibility that the act may not succeed. For example, suppose that I sincerely request you to open the door. The goal of my request is that you open the door and the intention of my request is that it be mutually believed that I want you to open the door. My request is successful if you recognize that I want you to open the door and my request is satisfied if you actually open the door in response to my request. The best I can do is to make my intention known to you and it is up to you whether or not you actually open the door. If I have reason to believe that you have not properly understood my intention then I may repeat my request, that is, I may attempt again to make my intention known to you. Accordingly, attempt is defined as having a goal and an intention.

Definition 2.12. Attempt

$$\begin{aligned}(\text{ATTEMPT x } e\ p\ q\ \text{t}) \triangleq\ & \text{t?}; [(\text{BEL x } \neg p) \wedge (\text{GOAL x } (\text{HAPPENS } e; \Diamond p?)) \wedge \\ & (\text{INTEND x t?}; e; q?\ (\text{GOAL x } (\text{HAPPENS } e; \Diamond p?)))]?; e\end{aligned}$$

An attempt at time t to achieve p via q is a complex action expression in which the agent x is the actor of event e and just prior to e, the actor chooses that p should eventually become true, and intends that e should produce q relative to that choice. So, p represents some ultimate goal that may or may not be achieved by the attempt, while q represents what it takes to make an honest effort. If the attempt does not achieve the goal p, the agent may retry the attempt, or try some other strategy or even drop the goal. However,

if the attempt does not succeed in achieving the intended effect q, the agent is committed to retrying until it is either achieved or impossible or irrelevant [107, 26, 27].

Compositionality is one of the basic characteristics of speech acts [26, 30, 4, 33, 97]. Accordingly, communicative acts based on a speech-act theory must have a composable semantics. We define two primitive communicative acts, REQUEST and INFORM, and all other communicative acts that we need are composed using these basic communicative acts by either specializing their content or by composing them in a sequential fashion. The technique of composing new communicative acts from an existing set of well-defined primitive acts leads to a consistent and well-defined semantics and also offers the possibility that agents can create a new performative by composing primitive ones and understand a new performative by decomposing them into their primitive components. We define our primitive communicative acts as attempts and their definitions that follow are derived from [106, 107].

2.5.1 Basic communicative acts

We consider REQUEST and INFORM to be the basic primitive acts and we define all other communicative acts using these two communicative acts.

Definition 2.13. Request

(REQUEST x y e a q t) $\triangleq$ (ATTEMPT x e ϕ ψ t)
where ϕ = (DONE y a) $\wedge$ [PWAG y x (DONE y a) (PWAG x y (DONE y a) q)$\wedge q$]
and ψ = (BMB y x (BEFORE e [GOAL x (AFTER e (BEL y [PWAG x y ϕ q]))])))

This definition of REQUEST requires the requesting agent to notify the other agent should he change his mind or discover a problem with the REQUEST. This requirement follows from the persistent weak achievement goal (PWAG x y ϕ q) that the requestee will have after performing the request. By communicating this PWAG, the requester is already treating the requestee as a teammate [106]. The goal of a REQUEST is that the requestee y eventually does the action a and also has a PWAG with respect to the requester x to do a. The requester's PWAG is relative to some higher-level goal q. The requestee's PWAG is not only with respect to the requester's PWAG towards her that she does the action a but also relative to the requester's higher-level goal q. The intention of

the request is that the requestee y comes to believe there is a mutual belief between the requestee and the requester that before performing the request, the requester had a goal that after performing the request, the requestee will believe that he (the requester) has a PWAG towards requestee to achieve the goal ϕ of the request.

Theorem 2.2. *Successful performance of the REQUEST communicative act establishes a mutual belief by default between the requester and the requestee about the requester's PWAG towards the requestee for doing the requested action. Formally,*

$\models$ (DONE (REQUEST x y e a q t)) $\Rightarrow$ (MB x y (PWAG x y ϕ q))
where, ϕ = (DONE y a) $\wedge$ [PWAG y x (DONE y a) (PWAG x y (DONE y a) q)$\wedge$ q]

Proof. The intention of a REQUEST is to bring about ψ in Definition 2.13. It is given that the REQUEST was successful. Therefore, the intention part of the REQUEST must have been achieved, that is,

(1) (BMB y x (BEFORE e [GOAL x (AFTER e (BEL y [PWAG x y ϕ q]))]))

The requester is assumed to be sincere, that is,

(2) (SINCERE x y (PWAG x y ϕ q)) [Proposition 2.5]

Therefore, we can conclude that

(3) (BMB y x (BEFORE e [GOAL x (AFTER e (KNOW y [PWAG x y ϕ q]))]))

[(1) & Definition 2.9]

That is, requester x had a goal that y come to *know* that [PWAG x y ϕ q]. Therefore, after e, agent x must have that PWAG himself. Consequently, x must believe it himself.

(4) (BEL x (PWAG x y ϕ q))

It is given that y does not have any belief to the contrary, that is,

(5) (BMI y x $\neg$(PWAG x y ϕ q)) [Assumption 2.2(a)]

Therefore, we can conclude that

(6) (BMB y x (PWAG x y ϕ q)) [Proposition 2.6]

The sender believes that the recipient does not have any belief to the contrary. That is,

(7) (BEL x (BMI y x ¬(PWAG x y ϕ q))) [Proposition 2.8]

Therefore, the sender believes that the recipient believes that it is mutually believed that (PWAG x y ϕ q).

(8) (BEL x (BMB y x (PWAG x y ϕ q))) [(7)&Proposition 2.7]

From (4), (8), (6) and Definition 2.3 of BMB and MB, we can conclude that,

(MB x y (PWAG x y ϕ q))

□

Definition 2.14. Inform
(INFORM x y e p t) $\triangleq$ (ATTEMPT x e ϕ ψ t)
where ϕ = (BMB y x p) and
ψ = (BMB y x (BEFORE e [GOAL x (AFTER e [BEL y (BEFORE e (BEL x p))])]))

The goal of an INFORM is that the listening agent y come to believe that there is mutual belief between him and the informing agent x that the proposition p is true. The intention of the INFORM is that the listening agent come to believe that there is mutual belief between him and the informing agent that, before performing the INFORM, the informing agent had the goal that after the INFORM is performed, the listening agent will believe that the informing agent believed p before performing the INFORM.

Theorem 2.3. *Successful performance of the INFORM communicative act establishes a mutual belief by default between the sender and the listener, that the sender believes the informed proposition. Formally,*

$\models$ (DONE (INFORM x y e p t)) $\Rightarrow$ (MB x y (BEL x p))

Proof. The INFORM was successful. Therefore, the intention part of the INFORM must have been achieved, that is,

(1) (BMB y x (BEFORE e [GOAL x (AFTER e [BEL y (BEFORE e (BEL x p))])]))

The requester is assumed to be sincere, that is,

(2) (SINCERE x y (BEFORE e (BEL x p))) [Proposition 2.5]

Therefore, using (1) and Definition 2.9 we can conclude that

(3)

(BMB y x (BEFORE e [GOAL x (AFTER e [KNOW y (BEFORE e (BEL x p))])]))

That is, sender x had a goal that y come to *know* that (BEFORE e (BEL x p)). Therefore, x must believe it himself.

(4) (BEL x (BEFORE e (BEL x p)))

The performance of the INFORM communicative act does not change the truth-value of (BEL x p).

(5) $\neg$(DONE (BEL x p)?; e; $\neg$(BEL x p)?) [Assumption 2.2(b)]

Therefore, if x believed p before the communicative act was done, it must still believe p.

(6) (BEL x (BEL x p)) [(4) and (5)]

It is given that y does not have any belief to the contrary, that is,

(7) (BMI y x $\neg$(BEFORE e (BEL x p))) [Assumption 2.2(a)]

Therefore, we can conclude that

(8) (BMB y x (BEFORE e (BEL x p))) [Proposition 2.6]

The communicative act does not change the truth-value of (BEL x p). Therefore, we can conclude that

(9) (BMB y x (BEL x p)) [(8), (5), & Proposition 2.4]

The sender believes that the recipient does not have any belief to the contrary, that is,

(10) (BEL x (BMI y x $\neg$(BEL x p))) [Proposition 2.8]

Therefore, the sender believes that the recipient believes that it is mutually believed that (BEL x p).

(11) (BEL x (BMB y x (BEL x p))) [(10), & Proposition 2.7]

From (6), (11), (9) and Definition 2.3 of BMB and MB, we can conclude that,

(MB x y (BEL x p))

□

From the above theorem, we can conclude that a single INFORM from agent x to agent y that p is not sufficient to establish mutual belief that p, rather the mutual belief that x believes p. However, in the special case when it is mutually believed that agent x is competent about p, a single inform will establish the mutual belief that p. In general, it takes an INFORM followed by a confirmation by the other agent that the informed proposition was believed, to establish mutual belief that p. We state these in the following lemmas.

Lemma 2.2. *Successful performance of an INFORM communicative act that p establishes mutual belief that p by default if it is mutually believed that the sender is competent with respect to p, that is, if (MB x y ((BEL x p) ⊃ p)). Formally,*

$$\models (\text{DONE (INFORM x y } e\ p\ t)) \wedge (\text{MB x y } ((\text{BEL x } p) \supset p)) \Rightarrow (\text{MB x y } p)$$

Proof. From the premise, we can conclude that

(MB x y (BEL x p)) ∧ (MB x y ((BEL x p) ⊃ p))	[By Theorem 2.3]
⊃ (MB x y (BEL x p) ∧ ((BEL x p) ⊃ p))	[⊨ (MB x y p) ∧ (MB x y q) ⊃ (MB x y $p \wedge q$)]
⊃ (MB x y p)	[Application of Modus-Ponens]

□

If ϕ is valid in our model (i.e., if ϕ is a theorem), then it is mutually believed that ϕ. As such, Lemma 2.2 is particularly useful for those propositions p that have the competence property that

$$\models (\text{BEL x } p) \supset p$$

From Lemma 2.1, PWAG is one such proposition, because

$$\models (\text{BEL x (PWAG x y } p\ q)) \supset (\text{PWAG x y } p\ q)$$

Therefore, an INFORM that the sender has a PWAG towards the recipient establishes mutual belief that the recipient has that PWAG, without the need for a separate confirmation from the recipient.

Lemma 2.3. *Successful performance of an INFORM that p followed by a successful confirmation by the recipient of the INFORM that it believes p establishes mutual belief that p, by default. Formally,*

$\models$ (DONE (INFORM x y e p t);(INFORM y x e_1 (BEL y p) t_1)) $\Rightarrow$ (MB x y p)

Proof. By applying Theorem 2.3 to each INFORM in the premise, we can conclude that

(MB x y (BEL x p)) $\wedge$ (MB x y (BEL y (BEL y p)))
$\supset$ (MB x y (BEL x p)) $\wedge$ (MB x y (BEL y p)) [Belief introspection]
$\supset$ (MB x y p)
[Re-writing MB in terms of BMB and re-arranging terms using Definition 2.3]

□

Next, we define the composed communicative acts AGREE, REFUSE, and CANCEL.

2.5.2 Composed communicative acts

AGREE and REFUSE are composed primitive acts defined from INFORM with specialized content. These communicative acts are used in the Request and the Standing Offer conversation protocols. We will show in Theorem 2.5 that a REQUEST followed by an appropriate AGREE is sufficient to create the inter-locking PWAGs, and hence the JPG required to form a team.

Definition 2.15. Agree
(AGREE[10] x y e a q t) $\triangleq$ (INFORM x y e ϕ t)
where ϕ = (PWAG x y (DONE x a) (PWAG y x (DONE x a) q) $\wedge$ q)

An agreeing agent x informs the listening agent y that he has a PWAG with respect to y to perform action a with respect to both y's PWAG that x do a relative to q, and q.

Theorem 2.4. *Successful performance of an AGREE communicative act establishes mutual belief by default that the sender x has the specified PWAG towards the recipient y. Formally,*

$\models$ (DONE (AGREE x y e a q t))
$\Rightarrow$ (MB x y (PWAG x y (DONE x a) (PWAG y x (DONE x a) q)$\wedge q$))

[10] AGREE was previously called CONFIRM in [106, 107]

Proof. It is given that

(1) (DONE (INFORM x y e ϕ t))

where, ϕ = (PWAG x y (DONE x a) (PWAG y x (DONE x a) q)$\wedge q$)

Recall that, for this ϕ

(2) $\models$ (BEL x ϕ) $\supset$ ϕ [From Lemma 2.1]

Therefore, we have

(3) (MB x y ((BEL x ϕ) $\supset$ ϕ))

[If ϕ is valid in our model, then it is mutually believed that ϕ]

Hence, we can conclude that

(MB x y ϕ) [From (1), (3), and Lemma 2.2]

□

The agreeing agent x believes that it has a PWAG towards agent y, and therefore, from Lemma 2.1, it in fact has that PWAG. A PWAG cannot be dropped unless there is appropriate mutual belief between the agent who has the PWAG and the agent towards whom the PWAG is directed. Therefore, if the agreeing agent x ever changes its mind about its PWAG towards agent y, then it needs to first establish a mutual belief with agent y.

Theorem 2.5. *A REQUEST immediately followed by AGREE in response to the request establishes a joint persistent goal (JPG) between the initiator and the participant, assuming instantaneous communication. Formally,*

$\models$ (DONE (REQUEST x y e a q t);(AGREE y x e_1 a q t_1))

$\Rightarrow$ (JPG x y (DONE y a) (PWAG x y (DONE y a) q)$\wedge q$)

Proof. This is a restatement of Theorem 8.2 that appears later in this dissertation. See Theorem 8.2 for a detailed proof. □

Definition 2.16. Refuse

(REFUSE x y e a q t) $\triangleq$ (INFORM x y e ϕ t)

where $\phi = \Box\neg$(PWAG x y (DONE x a) (PWAG y x (DONE x a) q)$\wedge q$)

A refusing agent informs the listening agent that he will never[11] have the PWAG to perform action a with respect q and with respect to y's PWAG that x do a relative to q. The effect of a REFUSE is opposite to that of the AGREE. A CANCEL communicative act is the counterpart of REFUSE that allows a requester to withdraw its PWAG towards the requestee. However, a CANCEL is permitted only if the requestee has not already communicated his PWAG towards the requester (for example, if the requestee has not yet agreed the requester's request).

Definition 2.17. Cancel

$$(\text{CANCEL x y } e\ a\ q\ t) \triangleq \eta?;(\text{INFORM x y } e\ \phi\ t)$$
$$\text{where} \quad \phi = \neg(\text{PWAG x y (DONE y } a)\ q),$$
$$\text{and} \quad \eta = (\text{EARLIER (PWAG x y (DONE y } a)\ q)) \wedge$$
$$\neg(\text{PWAG y x (DONE y } a)\ (\text{PWAG x y (DONE y } a)\ q) \wedge q)$$

A CANCEL communicative act is an INFORM that the initiator does not have a PWAG towards the participant to do a relative to q in the context of an earlier PWAG (whose cancellation is being informed). The canceling of a request by the initiator (if the initiator was the requester in an earlier interaction) allows the participant to drop his PWAG towards the initiator if the participant's PWAG is relative to the initiator's PWAG. However, as mentioned earlier, a request cannot be cancelled by the initiator if the requestee has already agreed to it – in this case, the resulting JPG needs to be discharged properly.

2.6 SUMMARY

We reviewed the logical framework of joint intention theory, presented definitions of basic concepts, and established some important results that will be needed in the remaining chapters. The notion of PGOAL represents individual commitment of an agent, and PWAG represents the commitment of an agent directed towards another agent. Two agents have interlocking PWAGs towards each other are bound together by a JPG (or joint commitment). Regarding communication, REQUEST and INFORM are the basic

[11] This is a simplification for the propose of this dissertation.

communicative acts in this framework. They are used to establish mutual beliefs and joint commitments (via establishment of mutual belief about interlocking PWAG). Next, we introduce STAPLE as a language for programming teamwork and communication based on the joint intention theory discussed in this chapter.

Chapter 3

Using STAPLE for Programming Teamwork and Communication

We introduce a declarative agent programming language called STAPLE (Social and Team Agents Programming Language) and use it to program multi-agent systems that exhibit automatic team and communicative behavior. In some aspects, STAPLE is similar to collaboration infrastructures such as Collagen [93] that support collaboration as well as dialogue based on a formal theory of teamwork. Theoretical differences apart, STAPLE is a more declarative general purpose agent programming language in that it supports first principles reasoning about teamwork and communicative acts, whereas Collagen gets its collaborative and dialogue behavior by implementing the algorithms and discourse structures of the Shared Plans theory [46]. Similarly, STAPLE shares its planning terminology and constructs as well as stack-based tracking of commitments and intentions with PRS [43]. However, PRS is intrinsically a single agent system with no built-in support for teamwork. As such, the support for first principles reasoning about teamwork and communication in STAPLE distinguishes it from PRS.

The basic logical constructs of joint intention theory are built into STAPLE and these constructs are directly interpreted by the STAPLE interpreter. The usual agent programming constructs such as strategies, policies, rules of rational action, and action definitions are independent declarative concepts. An agent specification in STAPLE consists of initial mental state (beliefs, goals, commitments, and intentions), capabilities (actions and plans that the agent is capable of performing), inference rules, and domain encoding (initial state of the world, domain specific inference rules, etc.). We present an overview of

STAPLE in Section 3.1, and describe a "Lights World" domain in Section 3.2 that is used for examples in this chapter. In Section 3.3, we discuss how to program single agents in STAPLE and in Section 3.4 we illustrate team and communicative behavior using a series of examples. We will revisit these examples later chapters to show why the agents behaved as per predictions of joint intention theory.

3.1 OVERVIEW OF STAPLE PROGRAMS

We first review the actual syntax of the language and then present an example of a STAPLE agent. The language elements presented here are concrete representations of the terms and constructs from joint intention theory described in the previous chapter.

3.1.1 Representing the Terms and Constructs

Agents in STAPLE are programmed using a Prolog syntax extended by operators for dynamic logic of actions (concurrent actions, test, repetition, etc.), temporal logic (eventually and always), and for negation, implication, conjunction, and some other miscellaneous constructs. These operators are chosen to resemble operators in the logical language of Joint Intentions as far as possible without having to redefine Prolog's built-in operators. For instance, $\Diamond$(BEL x $\Box\neg p$) is written as `<>bel(x,'[]'~p)` and (INTEND x a|b* p$\wedge$q) as `intend(x,(a;b..),p/\q)`. Variables in STAPLE start with upper case letter similar to Prolog variables. There are three constructs in STAPLE that warrant special mention: actions, plans, and rules.

Actions

In logic, it suffices to abstract away from details of an action by representing them by letters such as "a" and defining functions to give the preconditions that must be true before the action can be executed, and post-conditions that must be true after successful execution of the action. These functions can then be incorporated into the logic itself using some notion of causality[1], for instance, by saying that, if it is always the case that

[1]There is no notion of causality in this logic per se. Also, the suggested definition of effect that follows gets into the frame problem – it requires that no other event should happen concurrently with action a that

a certain proposition p becomes true whenever action a has just been done, then it is reasonable to treat p as an effect of the action a.

$$p = \mathit{effect}(a) \text{ iff } \models \Box[(\text{DONE } \neg p?; a) \supset (\text{DONE } \neg p?; a; p?)]$$

The actions in an actual program, however, have to be executable, and suitable for reasoning. As such, STAPLE requires all primitive actions to have at least two components - an executable component or the "code", and an "effect" that must be a proposition[2]. The effect of an action is defined as a proposition that always becomes true after performance of the action. Primitive actions may optionally specify a precondition (that must be true before the action can be executed), a context (that must be true in the beginning and must remain true throughout the action execution), and a list of desired effects (that may eventually become true as a result of the action being performed but there is no guarantee that they will ever become true).

Agent programmers have the flexibility to define actions in either Prolog or Java. In either case, there is no restriction on the code for an action – it can be any arbitrary piece of program but the corresponding effect for that action must express in logic what becomes true when the code for the action is executed (other than the fact that the given action was just done). So, from the point of view of the STAPLE interpreter, primitive actions are at the lowest level of reasoning granularity and can be treated as atomic actions for most purposes. Execution of these actions can either succeed or fail, and their effect is assumed to be true when they succeed.

No assumption is made about the time that a primitive action takes to complete. As mentioned before, it should be possible to "agentify" any legacy program using this representation of actions – agent programmers need only to specify the effects for legacy programs at a level at which they want the STAPLE interpreter to reason about them, and tell the STAPLE interpreter how to execute the code associated with those effects. In

---

may affect the truth value of the proposition that is being considered to be the effect of a. Nevertheless, this example is still a useful approximation of causality for the purpose of this dissertation.

[2]More accurately, actions in STAPLE can have a list of alternative effects whose probabilities add up to one. This representation provides another place to integrate statistical reasoning methods into the current logical framework (the importance value associated with commitments, rules, etc. is the other location). However, we will assume actions to have just a single effect with a probability of 1.0 in this dissertation.

fact, it is conceivable even to treat non-team aware agents as part of teams by using proxy agents that reason using this representation of actions, and to be able to prove offline that the non-team aware agents do in fact appear to behave as team members. STAPLE actions can have a few other optional convenience components that we will ignore in this dissertation.

Plans

Plans in STAPLE are named action expressions. The action expression of a plan is hierarchically composed from primitive actions and other existing plans using the action formation operators for sequence, non-deterministic OR, concurrent action, test action, and indefinite repetition. Further, because plans are also actions (though not singleton actions), they are required to specify the effects that must be true after a successful execution of the action expression for that plan. The STAPLE interpreter presented in this dissertation does not compute the overall effect of a plan from its constituent actions, and therefore, it must be computed offline by the agent programmer using techniques such as those of [94]. Note that there can be more than one way to execute a plan successfully, for instance, when the action expression for the plan consists of a non-deterministic OR. However, plans do not have a code, rather they have a "body" which is the complex action expression for the plan. Just like primitive actions, plans may optionally specify a precondition, a context, and some other convenience terms.

Rules

The declarative rules[3] in STAPLE can either be inference rules, or reactive rules, or rules for rational action, and are specified as P==>Q where both the antecedent P and consequent Q can be a conjunction of propositions. These rules tell a STAPLE agent what to do when a certain condition is satisfied but the joint intention theory leaves room for the agents to decide upon the best course of action consistent with the theory. For

[3]Note that these are rules in a STAPLE program as opposed to the defeasible rules of communication discussed in the previous chapter on logical background. One way to implement the defeasible rules of communication is to encode them as STAPLE rules.

example, one of the rules in STAPLE specifies that if an agent is committed to achieving a proposition, and it believes that there are actions that it can do to achieve that proposition, then the agent will intend to perform a non-deterministic OR expression of those actions with respect to the original commitment [98]. Rules are ordered according to a notion of importance and this makes it possible for agent programmers to override the default STAPLE rules using their own rules.

We now present a concrete example of a STAPLE agent, along with examples of actions, plans and rules.

3.1.2 A STAPLE Agent

An agent specification in STAPLE consists of the agent meta-information (such as the name and address of the agent), and its beliefs, goals, and commitments along with any plans, actions, and rules specific to that agent. The STAPLE default rules, the action library, the plan library, and the protocol library are accessible to all agents. Table 3.1 shows the example of a STAPLE agent using the actual syntax. This example agent is based on a "Lights World" domain similar to that used by [13] to demonstrate human-robot collaboration. In this domain, there are three lights that the human and a robot are to turn on or off collaboratively. Breazeal's domain is discussed further in Section 3.2.

The agent in Table 3.1 believes that there are three lights in the world `(redlight, bluelight,` and `greenlight)` and that none of these lights are initially switched on. This agent has a commitment (PGOAL) for achieving a state `lights_are_on` and this PGOAL is with respect to its belief that it is dark outside. This agent program defines a plan called `turn_lights_on` that takes no argument and does not define any precondition. The effect of this plan is that eventually all the lights will be on, that is, <>`lights_are_on` and its body is the action expression $(p?;a)^*|\neg p?$, that is, test proposition p and perform action a if the test succeeds and do this indefinitely until the test for p fails. This is the same as saying "while p do a" in a traditional programming language where p is the proposition `bel(self,`$\sim$`switched_on(X))` and a is the action `switch_on(X)`. The test for $\neg p$ is slightly different from the test for p – specifically, it is the test for `(BEL self` $\exists$`x` $\neg$`switched_on(x))` where we have used the existential quantifier so that the variable x in

Table 3.1: Example of a STAPLE Agent Program

```
% Agent meta-information
agent_name(self, simagentx).
observes(world_sim(lightdomain,all)).

% Agent's beliefs, Domain encoding, Initial state of the world
bel(self,light(redlight)).
bel(self,light(bluelight)).
bel(self,light(greenlight)).

bel(self,~switched_on(redlight)).
bel(self,~switched_on(bluelight)).

bel(self,~switched_on(greenlight)).
bel(self,dark(outside)).

% Agent's commitments and intentions
% The number 8.0 specifies the importance of the pgoal
pgoal(self, lights_are_on, bel(self,dark(outside)), 8.0).

action_definition(switch_on,1) :-
    [args:  [Light],
    precondition:  {bel(self, light(Light)∧(~switched_on(Light)))},
    code:  {world_sim(do_action(lightdomain,switch_on(Light)))},
    effects:  [(switched_on(Light),1.0)]
    ].

% Body of this plan is the following action expression in logic:
% WHILE(bel(self,~switched_on(X))) DO (switch_on(X))
% Also, its effect is that eventually lights are on.
plan(turn_lights_on, 0) :-
    [body:  {((bel(self,~switched_on(X))?, switch_on(X))..)  ;
              ~bel(self,exists(X,~switched_on(X)))?
          },
    effects:  [(<>lights_are_on,1.0)]
    ].

% Effects of the above plan depends on the effects of its component actions
bel(self,lights_are_on) :- istrue(~bel(self,exists(X,~switched_on(X)))).

% Rational rules available to this agent
rule(rational1, pgoal(self, P, Context, Imp), 5, StackId) :-
    findall(Action,bel(self,can_achieve(P,Action)),Actions),
    \+ Actions = [],
    NewContext = pgoal(self, P, Context, Imp),
    ==>
    subgoal(pgoal(self,done(self,or(ActionList)),NewContext,Imp),StackId).
```

$\neg p$ does not get bound by tests for p. The action `switch_on` is defined in Table 3.1 as an action that takes a light as an argument and whose precondition is that the agent believes that argument of the action is in fact a light and that it is not switched on. The code for this action executes an instruction to switch on the light and its effect is that the light is switched on, that is, `switched_on(Light)` with probability 1.0.

The rule `rational1` at the bottom of Table 3.1 has an importance 5 and it is applicable only when the agent has a PGOAL to achieve a proposition – it says that if the agent has a commitment to achieve a proposition p and the agent can find a non-empty list of actions that it believes can achieve p then the agent will have a new commitment for doing the non-deterministic OR of those actions relative to the original commitment. Note that the first conjunct of the rule's antecedent is treated as a precondition of the rule, and is specified as part of the rule preamble to help quickly narrow down the list of rules applicable in a give situation.

3.1.3 Communicative acts in STAPLE

STAPLE treats communicative actions just like any other action. As such, they must have at least code and an effect. The code of a communicative action for artificial agents simply creates a message and sends it off to the recipient over the network. The effect of a communicative action is set to be the *intention* part in the semantics of the communicative act defined as an attempt (Section 2.5). As mentioned earlier, STAPLE actions may optionally have a list of desired effects. The desired effect of a communicative action is set to be the *goal* part in the semantics of that communicative act and it is used by the STAPLE interpreter for action selection during means-end reasoning.

We assume that the performance of communicative actions does not change the proposition being communicated (Assumption 2.2b). Therefore, if an agent believed p right before performing a communicative act that p then it also believes p right after performing that communicative act. We also assume that agents are sincere in their communication (Proposition 2.5). Its consequence is that if an agent performs a communicative act intending that the recipient come to believe p as a result of the communicative act then that agent must itself believe p. As a result of these assumptions, we can get rid of the

"BEFORE" and "AFTER" predicates in the intended and desired effects of the communicative acts in Section 2.5. Another simplification is that all facts in an agent's belief base are interpreted as if the agent believes those facts. Therefore, unilateral mutual belief (BMB – meaning that an agent believes there is mutual belief) predicate in the effect and desired effect attributes can be re-written as mutual belief (MB). The precondition of the request communicative act is that the requestee does not already believe the propositions about which the requester intends to establish mutual belief via the request. With these changes, the definition of REQUEST in Definition 2.13 translates to the following action definition in STAPLE (with four arguments).

Definition of Request:

```
action_definition(request,4) :-
[args:  [X,Y,A,Q],
code:  {% code to compose and send message },
precondition:  {~bel(X,done(A))∧
                ~bel(X, pwag(Y,X,done(A),pwag(X,Y,done(A),Q)∧ Q))∧
                ~bel(X, bel(Y,pwag(X,Y, done(A)∧pwag(Y,X,done(A),
                     pwag(X,Y,done(A),Q)∧Q),Q)))
                }
desired_effects:  [<>done(A),
                   <>pwag(Y,X,done(A),pwag(X,Y,done(A),Q)∧Q) ],
intended effects:  mb(X,Y, pwag(X,Y, done(A)∧pwag(Y,X,done(A),
                     pwag(X,Y,done(A),Q)∧Q),Q))
].
```

In the above definition, `X` is the actor of the request communicative act, `Y` is the recipient as well as the intended actor of the requested action `A`, and `Q` is the relativizing condition of the request. The other communicative acts in Section 2.5 are similarly simplified and defined in STAPLE programs. We will revisit these definitions in Chapter 4.

3.1.4 Other Aspects of the STAPLE Interpreter

The STAPLE interpreter decomposes action expressions by following their definition in dynamic logic. For instance, the intention to do an action sequence is interpreted as follows: If the agent believes that the first action of the sequence as not been done then it intends the first action relative to the original intention, and if it believes that an initial

subsequence of the action expression has been done then it intends the remainder of the sequence relative to the original intention. Recall that this consequence of intending an action sequence is a theorem of the logic discussed earlier (Section 2.3.1). The interpretation of joint action expression is also similar except that agents may need to coordinate their actions using communicative acts, may need to decide the next action in a joint OR, and may need to decide the actor(s) of the next action(s).

The STAPLE interpreter includes a Horn-clause belief reasoner that implements weak S5 semantics and is capable of reasoning with quantified beliefs. For tractability reasons, it uses a controlled search space, imposes syntactic limitations on the structure of terms in the belief base, and requires belief terms to be reduced to a simpler form before asserting into the belief base. A discussion on the belief reasoner as well as other components of the STAPLE interpreter appears in Chapter 4. An interesting behavior of the STAPLE interpreter is when it discovers an unbound variable in the action that it is committed to doing. It adopts a commitment for finding the value of that variable, that is, a PGOAL for KNOW-REF [1]. This behavior is governed by a declarative rule and an example of its usage is discussed in the following sections.

Next, we discuss the domain and setup for the examples that follow. The simulator for this domain is not part of STAPLE. It simply provides a programmatic interface to test STAPLE and to demonstrate its capabilities.

3.2 LIGHTS DOMAIN AND THE EXPERIMENT SETUP

The examples presented in this dissertation are based on a "Lights World" domain similar to that used by Breazeal et. al. [13] to demonstrate human-robot collaboration using joint intention theory. In this domain, there are three lights that the human and a robot are to turn on or off collaboratively (Figure 3.1). The teammates engage in turn-taking, recover from problems, and communicate via gestures such as head nods and facial expressions. Breazeal and colleagues use a goal-driven hierarchical task representation where the task at each level in the hierarchy is associated with a goal. The goals may be state-change goals or "just-do-it" goals that are not concerned with achieving state changes. In accordance with joint intention theory, the robot maintains a shared mental state with the human, demonstrates a commitment to doing its own part of the joint action as well as the action to be done by its teammate, and communicates to establish the required mutual beliefs for the jointly committed goal.

Figure 3.1: The Robot and Lights in Breazeal et. al. [13]

This domain is particularly interesting for our experiments because it has already been programmed successfully using joint intention theory, and because it allows several variations such as what a robot can observe, what are the mutual beliefs of the teammates, and what the collaborative joint action is that the team is supposed to jointly execute. Compared to Breazeal's implementation, STAPLE does not distinguish explicitly between achievement goals and goals to do an action as separate types because they are expressed directly in the dynamic logic that is part of joint intention theory. Instead of simple hierarchical task representation, STAPLE uses plans that can be arbitrarily complex action expressions. Breazeal's implementation does consider timing of actions, which STAPLE agents do not reason about yet. One benefit of STAPLE is that it allows a user to specify and/or modify team behavior declaratively, thereby significantly short-cutting development time. This approach contrasts with Breazeal's framework where a new algorithm has to be programmed into the robot to get the new collaborative behavior.

For our experiments, we use the "Lights World" simulator shown in Table 3.2 that simulates Breazeal's button pressing task setup [13]. The simulator has three lights (red, blue, and green) and supports two actions: `switch_on`, and `switch_off` each of which take the name of the light as an argument. The simulator can be configured so that an agent can observe everything that is going on in the world such as what is the current state of the lights, who switched on or off various lights, and whether or not an action was successful. It can also be set up so that an agent can only observe the effects of its own actions and cannot see anything else. The simulator allows external agents (such as the experimenter) to manipulate the lights manually, a feature that can be used to introduce unexpected problems into any jointly intended task.

The STAPLE agents in our experiment are attached to the simulator (i.e., they can observe the simulated world via their sensors and perform actions on it via their effectors).

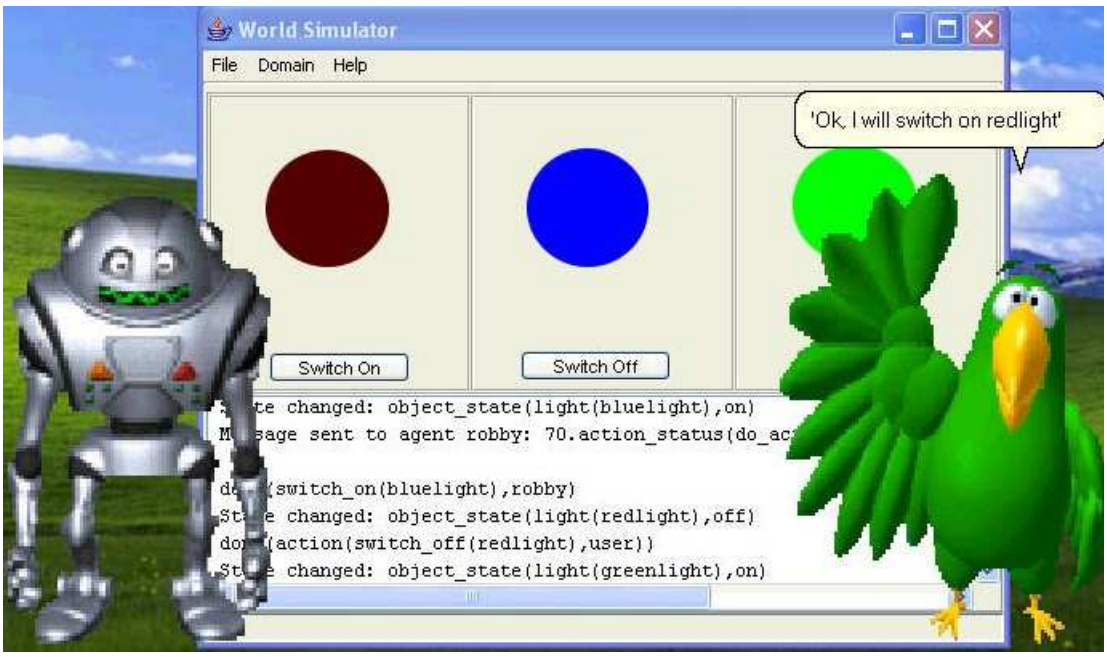


Figure 3.2: A snapshot of Lights World simulator

The agents communicate directly with each other using messages based on the communicative actions that we define. A template based natural language generator is connected to the agents in order to translate communicative act representations into natural language. Each STAPLE agent is also connected to a separate Microsoft animated agent character for the purpose of demonstration. Although not used for the current experiments, it is possible to have the animated characters use gestures and facial expressions, as done in [13] to deliver non-linguistic communication instead of natural language dialogue. Further, each STAPLE agent in our experiment has a GUI console that gives us direct access to its belief base through a Prolog interface and enables us to pause and resume the agent.

Next, we will use STAPLE agents for the experimental setup discussed above under different initial conditions.

3.3 SINGLE AGENT EXAMPLES

We will use the agent program in Table 3.1 to illustrate that the STAPLE agent `simagentx` behaves according to joint intention theory. When we run the program in Table 3.1, the agent registers with the lights world simulator to observe all the lights in the world as per the instruction in the agent meta-information. From this point onwards, the simulator

will act as a sensor and notify the agent of any action performed on the lights, the agent who performed the action, and the new state of the light upon which the action was performed. As a result of registering with the simulator, the agent also gets access to certain instructions that can be used to define actions accessible to the agent. For example, we have used one such instruction to define the action `switch_on`. This action sends an instruction to the simulator to turn on a specified light. We add another action `switch_off` (not shown in Table 3.1) similar to the action `switch_on` to the agent program before running it.

After the initial setup, the agent acts on its commitment (PGOAL) to achieve a state where lights are on. Recall from Definition 2.1 that an agent having a PGOAL to achieve p will keep that PGOAL at least until it believes that p has been achieved or is impossible to achieve or is irrelevant. Next, we will explore the behavior of the agent under these different discharge conditions for an individual commitment.

3.3.1 Committed goal is achieved

The agent is committed to achieving a state `lights_are_on` relative to its belief that there exists some light that is not switched on. We observe that all the three lights in the Lights World simulator turn on one by one. Again, this behavior is as predicted by the JI theory when used with the following rule of rational action (taken from Table 3.1).

```
rule(rational1, pgoal(self, P, Context, Imp), 5, StackId) :-
    findall(Action,bel(self,can_achieve(P,Action)),Actions),
    \+ Actions = [],
    NewContext = pgoal(self, P, Context, Imp),
    ==>
subgoal(pgoal(self,done(self,or(ActionList)),NewContext,Imp),StackId).
```

This rule says that, if the agent has a PGOAL for achieving p and if it believes that there are certain actions that can achieve p, then it will commit to perform a non-deterministic OR of those actions. A trace of the agent program shows that the agent acts on its commitment by looking for an action or action expression (i.e., a plan) that can potentially achieve the committed goal. It finds one such action – the plan called `turn_lights_on` and commits to doing that action. Specifically, the agent executes the rule `rational1` in Table 3.1 and this rule results in the agent having a new PGOAL

relative to the earlier one for doing an OR action expression that has only one action (the action `turn_lights_on`). This PGOAL is relative to the earlier PGOAL as shown by the `NewContext` in rule `rational1`. The agent acts upon this new commitment by picking up one action from the OR expression and committing to doing that action relative to the larger commitment for doing the entire OR expression. In this case, since the OR expression has only one action, the agent commits to achieving that action relative to the larger commitment. This action itself happens to be a complex action expression (i.e., the body of the plan `turn_lights_on`) so the agent commits to doing this action expression relative to the commitment for doing the OR action expression.

The new action expression[4] that the agent is committed to doing is of the form $(p?;a)^*|\neg q?$ where $p?$ is the test action `bel(self,`$\sim$`switched_on(X))?`, a is the action `switch_on(X)`, and q is the test action $\sim$`bel(self,exists(X,`$\sim$`switched_on(X)))?`. The agent acts on this PGOAL by choosing one action from the OR expression and committing to doing that action relative to the PGOAL for doing $(p?;a)^*|\neg q?$. If it picks up the test action $\sim$`bel(self, exists(X, `$\sim$`switched_on(X)))?` and commits to doing it, then it also intends to do that test action (due to another rule for rational action not shown in Table 3.1 which says that if the agent is committed to doing an action then it will intend to do that action if the new intention does not conflict with any existing commitment or intention). The agent executes that test action which fails because there are three lights that the agent believes are not switched on. At this point, the agent can keep performing the test action until it believes that it is impossible to do the test action. One of the STAPLE rules (modifiable by the agent programmer) says that an intention to do a test action is impossible to achieve if the test action fails. Using this rule, the agent decides that its intention for performing the test action is impossible and therefore, it drops that intention. By the same token, it drops the commitment for doing that test action because it is impossible. At this point, the agent is back to the larger PGOAL of doing the action expression $(p?;a)^*|\neg q?$ with the knowledge that the second action $(\neg q?)$ has failed once. So it picks one action from the remaining actions in the OR expression. In this case, the

[4]Recall from Section 2.2.4 that (WHILE p DO a) is defined as $(p?;a)^*|\neg p?$ and therefore, this action expression is basically the following while loop:

`WHILE bel(self,exists(X,`$\sim$`switched_on(X))) DO switch_on(X)`

However, instead of $(p?;a)^*|\neg p?$ we use a variation $(q?;a)^*|\neg p?$ where q is same as p but without the existential quantifier, that is, q is `bel(self,`$\sim$`switched_on(X))`. The reason for this is that the variable X in p is not a free variable as it is existentially quantified but we want to use a free variable in the test condition of the while loop so that this variable gets bound when the test is performed. The value that X gets bound to is used in the action `switch_on(X)`.

only action remaining is $(p?;a)^*$ so the agent commits to doing this action (expression) relative to the commitment for doing $(p?;a)^*|\neg q?$. Note that the formal semantics of non-deterministic OR in Chapter 2 says that the system knows somehow as to which branch to pick when it comes to an OR expression (as if it is told by an Oracle which branch to choose). In this case, the Oracle would have told the system to pick the branch $(p?;a)^*$ instead of the branch $\neg q?$. So the knowledge that the test action $\neg q?$ has failed once should not prevent the agent from re-trying it in the future. However, the agent would need to decide at some point whether it is impossible to do the OR action expression. Therefore, STAPLE agents are provided with a programmer modifiable rule such as "if every choice in an OR action expression fails n times (say, twice) then assume that that it is impossible to perform that OR action expression".

As an aside, if we run the agent again and again, we see that in some runs, the agent picks up the action expression $(p?;a)^*$ first and commits to doing it relative to the commitment for doing $(p?;a)^*|\neg q?$. This behavior is an approximation to the semantics of non-deterministic OR – the agent randomly picks up one action from an OR action expression and, if that action fails, then it picks up another action from the remaining actions in the OR expression. However, the agent is free to retry a failed action in the future.

The agent acts upon its commitment for doing the indefinite repetition $(p?;a)^*$ by committing to do $p?;a$ relative to the larger commitment. Furthermore, the agent acts upon the new commitment for doing the action sequence $p?;a$ by committing to the first action in the sequence relative to the commitment for doing the entire sequence. As such, the agent now has a commitment for doing the test action `bel(self,~switched_on(X))?`. The agent acts on this commitment by intending to do this test action and then actually doing this action. This test action succeeds and binds the variable X with one of the lights that is not switched on (say, `redlight`). The intention to perform the test action and the commitment to do the test action are successfully discharged. So the agent now commits to doing the next action in the action sequence and similar to earlier reasoning, it intends to do the action `switch_on(X)` where X is bound to `redlight`. The agent acts on this intention by executing the action `switch_on(redlight)` that turns on the red light. Therefore, this intention and the commitment to do `switch_on(X)` are successfully discharged. The agent is back to the state where it has a commitment to do the indefinite repetition $(p?;a)^*$. The entire process from that point onwards is repeated until the agent has turned on all the three lights.

After all lights have been turned on, the agent again has a commitment to do the indefinite repetition $(p?;a)^*$ relative to its commitment for doing the action expression $(p?;a)^*|\neg q?$. Similar to earlier steps, the agent ends up with an intention to do the test action `bel(self,~switched_on(X))?`. However, this time the test action fails because there is no light that is not switched on. As above, the agent decides that its intention to do the test action is impossible to achieve and therefore, it drops that intention. Similarly, it drops the commitment for doing the test action. The agent reasons that, if it is impossible to do the first action in an action sequence then it is impossible to do the entire action sequence. Therefore, it drops the commitment for doing $(p?;a)$. From Chapter 2, the indefinite repetition $(p?;a)^*$ is simply the sequence $(p?;a);(p?;a)^*$. As such, the agent reasons that it is impossible to do $(p?;a)^*$ and therefore, it drops the commitment for doing $(p?;a)^*$. At this point, the agent is left with a commitment for doing $(p?;a)^*|\neg q?$ with the knowledge that the choice $(p?;a)^*$ has just failed. So the agent commits to do the action $\neg q?$ relative to the commitment for doing $(p?;a)^*|\neg q?$. It intends the test action `~bel(self,exists(X,~switched_on(X)))?` and executes it. The test succeeds because there is no light that is not switched on. As such, the intention to do that test action has been achieved and therefore, the agent drops that intention. Similarly, the commitment for doing `~bel(self,exists(X,~switched_on(X)))?` has been achieved and therefore, the agent drops that commitment as well. Now, the agent reasons that an OR action expression has been done if at least one action in the OR expression has been done. Therefore, the agent has successfully discharged its commitment for doing $(p?;a)^*|\neg q?$ and so it drops this commitment. Similarly, the agent drops all other commitments that lead to this commitment (such as the commitment for doing `turn_lights_on` and the commitment for achieving `lights_are_on`) as they have all been successfully discharged. At this point all the lights are switched on and the agent has no undischarged commitment or intention.

3.3.2 Interfering with the committed goal

Our experimental setup enables us to pause and resume a running agent and to query and update its belief base using a graphical user interface that accepts Prolog commands. We use this facility to interfere with the agent's commitment and observe that its behavior is consistent with that predicted by the JI theory.

Case 1: When the agent has switched on one of the lights, we manually switch on the remaining two lights using the on/off button on the simulator for each light. The agent

is notified by the simulator that those two lights are switched on. The agent uses the reasoning rule

`bel(self,lights_are_on) :- istrue(~bel(self,exists(X,~switched_on(X))))`.

in Table 3.1 to infer that its committed goal `lights_are_on` is already achieved and therefore it drops this commitment. A trace of the agent program shows that any commitment that is relative to this original commitment is also dropped. This chain is followed until all commitments and intentions that eventually resulted from the commitment to achieve `lights_are_on` are dropped.

Case 2: When the agent has switched on two lights, we manually switch off one of those two lights. The agent eventually switches that light back on. We keep switching off one or two lights that the agent has turned on and the agent goes back and turns them on until it successfully turns on all the lights. A program trace shows that if we interfere with the agent in the above manner after it has committed to doing the indefinite repetition $(p?;a)^*$, the agent eventually finds the light(s) that we manually switched off via the test action `bel(self,~switched_on(X))?` and then switches it back on. The agent does not give up its commitment to turn all lights on until it believes that the committed goal has been achieved, that is, `lights_are_on` becomes true.

3.3.3 Committed goal is impossible

According to the definition of PGOAL, an agent can drop its commitment if it believes that the committed goal is impossible to achieve. We test the behavior of the agent program in Table 3.1 after making it impossible to turn on all the lights.

Case 1: We configure the lights world simulator to simulate a faulty switch for one of the lights. So whenever the agent sends the command to turn on the green light, the simulator notifies that the action failed and that the green light is still switched off. We also add a rule to Table 3.1 that if the agent fails to do an action twice then it concludes that it is impossible to do that action. On running the program in Table 3.1, the agent turns on the red light and the blue light. However, it fails to turn on the green light and eventually prints that it is impossible to achieve the state `lights_are_on`. As discussed earlier, a stack trace shows that, after the test action `bel(self,~switched_on(X))?` binds variable X to `greenlight,` the agent commits to doing the action `switch_on(greenlight)`. Thereafter, it intends to do that action and executes that action. However, the action fails. The agent retries the action twice (as per the rule that we specified) and concludes that it is impossible to do the action `switch_on(greenlight)`. Therefore, it drops its intention to switch on

the green light and reconsiders its commitment to do the action `switch_on(greenlight)`. It has already concluded that it is impossible to perform that action and it does not know of any other way to do that action. So it concludes that it is impossible to achieve that commitment and so drops this commitment as well. Similarly, the agent concludes that it is impossible to do indefinite repetition $(p?;a)^*$ and drops that commitment. At this point, the agent reconsiders its commitment for the OR expression $(p?;a)^*|\neg q?$ and decides to try the other choice in the OR expression. It adopts a commitment to do the test action $\sim$`bel(self,exists(X,`$\sim$`switched_on(X)))`, and then intends and executes that action. The test action fails because there does exist one light (the green light) that is not switched on. The agent concludes that the test action is impossible and drops its intention and commitment for performing that action. At this point, the agent again reconsiders its commitment for the OR expression $(p?;a)^*|\neg q?$ with the knowledge that both branches of the OR expression have failed once. It then applies any rules that will lead it to conclude that it is impossible to perform this OR action expression. As mentioned earlier, we have given the agent a rule that says "if every choice in an OR action expression fails twice[5] then assume that that it is impossible to perform that OR action expression". So the agent cannot conclude that the OR action is impossible and it again chooses one branch of the OR expression and commits to that action. The process is repeated all over again until both branches of the OR expression have failed twice at which point the agent concludes that it is impossible to perform that action expression and drops the commitment to do it. The agent finally reconsiders its highest level commitment to achieve the state `lights_are_on` and infers that it is impossible to achieve that state because it knows of only one action (the plan `turn_lights_on)` that can achieve this state but it has already found that it is impossible to perform that action. So it decides that it is impossible to achieve its committed goal `lights_are_`on and therefore, drops that commitment.

Case 2: We pause the agent when it is in the middle of switching on the lights and add the following rule and fact to its belief base that enables it to conclude that it is impossible to achieve its committed goal:

```
bel(self,[]~lights_are_on) :- istrue(bel(self,exists(X,faulty(X)))).
faulty(greenlight).
```

[5]Two is just a number that we chose for demonstration purposes. An agent programmer may choose any number, say 5 times, etc. Furthermore, that number doesn't have to be a fixed number, rather it can potentially be computed depending on the application, the context, and other factors.

Upon resuming the agent, it concludes that it is impossible to achieve `lights_are_on` (i.e., $\Box\neg$`lights_are_on`) and therefore, it drops its commitment to achieve `lights_are_on`. It also drops all other commitments and intentions that it had adopted relative to this high level commitment because they become irrelevant when the high level commitment is dropped.

3.3.4 Committed goal is irrelevant

Recall from Table 3.1 that the agent's PGOAL is relative to its belief that it is dark outside (i.e., `bel(self,dark(outside))`). We pause the agent as it is in the middle of switching on the lights and retract the statement `bel(self,dark(outside))` from its belief base. Thereafter, we add $\sim$`bel(self,dark(outside))` to the agent's belief base and then resume the agent. The agent immediately concludes that its commitment to achieve `lights_are_`on is irrelevant and drops that commitment. It also drops all other commitments and intentions that it had adopted relative to this high level commitment because they also become irrelevant when the high level commitment is dropped. This behavior is as predicted from JI theory.

3.3.5 Reactive rule to adopt a new commitment

This example illustrates that STAPLE agent programs need not specify any initial intention or commitment. The agent will adopt commitments and intentions as necessary as long as it does not conflict with any existing commitment or intentions.

Case 1: We modify the program in Table 3.1 by removing the lines with `pgoal` and `intend`. We also remove the initial belief `bel(self,dark(outside))` and add a rule to the program that tells it to adopt a new high level commitment to achieve `lights_are_on` if the agent comes to believe that it is dark outside. The modified agent program is listed in Table 3.2 and it has the new reactive rule towards the end.

After we start the agent, it does nothing. Then we add `bel(self,dark(outside))` to the agent's belief base (using its GUI console) and observe that the lights turn on one by one. A program trace shows that as soon as we add the new belief to the agent's belief base, its reactive rule fires and the agent adopts a PGOAL to achieve `lights_are_on`. From this point onwards, its behavior is same as if we had specified that commitment initially to the agent as in Section 3.3.4.

Case 2: We modify the agent program in Table 3.1 to add an initial intention to save electricity. We also add an inference rule that the agent is not saving electricity if the lights are on. When we run the agent program and add `bel(self,dark(outside))` to the agent's belief base as in the first case, the agent does not adopt the commitment to achieve `lights_are_on`. A trace of the agent program shows that the reactive rule fired as in the first case but the agent believed that the effect of the new commitment conflicts with an existing intention with a higher importance and so it does not adopt the conflicting commitment.

However, joint intention theory does not prevent an agent from adopting conflicting commitments as long as the commitments are for different times. For example, if the agent has an intention to save electricity for today then it can adopt a commitment to turn lights on for tomorrow. The current implementation of the STAPLE interpreter does not automatically reason about temporal dependencies if the dependency is not explicitly specified. So the temporal dependencies would have to be explicitly specified in the committed goal or action and the corresponding inference rule would need to be given to the agent to enable it to infer that the apparently conflicting commitments are actually not in conflict because they apply to different days.

3.4 TEAMWORK AND COMMUNICATION EXAMPLES

The examples that follow demonstrate the feasibility of obtaining team-oriented dialogue without having to program it explicitly. The most interesting part of these experiments is that we get a large range of dialogue behavior from the STAPLE interpreter under different initial conditions. For the experiment, we create two STAPLE agents Bob and Harry, and give Bob an initial individual commitment (PGOAL) to establish a joint commitment (JPG) with Harry to jointly turn lights on. These two agents are executed in separate copies of the interpreter at the same time. The action expression for the composite action (or plan) `jointly_turn_lights_on` can be modified to obtain different joint behavior from the agents as we see next. In all the examples that follow, we set the agents to follow the policy of executing joint action expressions in lockstep (Section 2.3.2).

3.4.1 Jointly Executing an Action Expression

We set a scenario that roughly models a situation where both agents are talking on a cellphone with one agent standing outside observing the lights on a tower and another

Table 3.2: STAPLE Agent with Reactive Rule

```
% Agent meta-information
agent_name(self, simagentx).
observes(world_sim(lightdomain,all)).

% Agent's beliefs, Domain encoding, Initial state of the world
bel(self,light(redlight)).
bel(self,light(bluelight)).
bel(self,light(greenlight)).

bel(self,~switched_on(redlight)).
bel(self,~switched_on(bluelight)).
bel(self,~switched_on(greenlight)).

action_definition(switch_on,1) :-
    [args:  [Light],
    precondition:  {bel(self, light(Light)∧(~switched_on(Light)))},
    code:  {world_sim(do_action(lightdomain,switch_on(Light)))},
    effects:  [(switched_on(Light),1.0)]
    ].
% Body of next plan is the following action expression in logic:
% WHILE(bel(self,~switched_on(X))) DO (switch_on(X))
plan(turn_lights_on, 0) :-
    [body:  {((bel(self,~switched_on(X))?, switch_on(X))..)  ;
             ~bel(self,exists(X,~switched_on(X)))?
        },
    effects:  [(<>lights_are_on,1.0)]
    ].
% Effects of the above plan depends on the effects of its component actions
bel(self,lights_are_on) :- istrue(~bel(self,exists(X,~switched_on(X)))).

% Rational rules available to this agent
rule(rational1, pgoal(self, P, Context, Imp), 5, StackId) :-
    findall(Action,bel(self,can_achieve(P,Action)),Actions),
    \+ Actions = [],
    NewContext = pgoal(self, P, Context, Imp),
    ==>
    subgoal(pgoal(self,done(self,or(ActionList)),NewContext,Imp),StackId).

% Reactive rule to adopt a new high level commitment if the agent believes
% that it is dark outside
rule(rational2, true, 8, StackId) :-
    istrue(bel(self,dark(outside))),
    Context = bel(self,dark(outside)),
    ==>
    adopt(pgoal(self, lights_are_on, Context, 8.0)).
```

agent inside the control room with controls to manipulate lights. We setup the agents initially as follows: (i) Bob knows the initial state of the world that there are three lights (Redlight, Bluelight, and Greenlight), and that none of the three lights are switched on. Harry does not know about the initial state of the world. (ii) Bob can observe the state of the world including any changes in its state (such as a light being switched on or off). However, Harry can only observe the effects of its own actions. For instance, if Harry switched on the Redlight then it can see whether or not Redlight turned on but it cannot see the state of the other lights. (iii) Harry is the only agent capable of performing actions `switch_on/1` and `switch_off/1` that take the name of a light as parameter. (iv) We set the body of the plan to jointly turn lights on so that one agent does the test action `~switched_on(Light)?` to find a light that has not been switched on followed by another agent turning on that light. This sequence is repeated indefinitely until the first agent fails to find any light that is not turned on and is represented by the following action expression where variable X is bound to 'Bob' and Y is bound to 'Harry':

```
WHILE bel(X,~switched_on(L)) DOES(Y, switch_on(L))
```

Table 3.3 summarizes the above settings and shows the plan to jointly turn lights on.

Given its initial mental state, Bob is committed to establishing a joint commitment with Harry to jointly execute the above action expression where Bob does the test action and Harry does the action of switching on the light. As a result, Bob and Harry engage in a dialogue that has at least two interesting aspects. First, it follows directly from joint intention theory as discussed below. Second, it illustrates the range of dialogue behaviors that follow by declaratively changing the initial conditions. This rich and varied range of dialogues is apparent by comparing it with the dialogue in the next experiment (Section 3.4.2). The transcript of an actual dialogue between Bob and Harry follows.

Dialogue 1:

1. Bob: Let us jointly turn lights on.
2. Harry: I agree to jointly turn lights on.
3. Bob: Redlight is not switched on.
4. Harry: Ok.
5. <action: Harry pushes button; Redlight turns on>

Table 3.3: Setup where Bob is Observer and Harry is Actor

Bob:
(i) Initial state:
- There are three lights (redlight, bluelight, and greenlight)
- None of these lights are switched on.

(ii) Observation:
- Can see state of the entire world
- Can see all changes in the state of the world

(iii) Actions:
- Cannot perform actions switch_on and switch_off

Harry:
(i) Initial state:
- Does not know this initial state of the world

(ii) Observation:
- Cannot see state of the entire world
- Can only see effects of its own actions
- Cannot see other changes in the world

(iii) Actions:
- Can perform actions switch_on and switch_off

```
% Both agents have this plan.  Its body is the following action expression:
% WHILE bel(X,∼switched_on(L)) DOES(Y, switch_on(L))
plan(jointly_turn_lights_on,5) :-
[description:  'Plan for turning lights on jointly',
    args:  [X,Y,E1,E2,E3],
    body:  { ((action(test(bel(X,∼switched_on(L))),X,E1),
               action(switch_on(L),Y,E2))..)  ;
               action(test(∼bel(X,exists(L,∼switched_on(L)))),X,E3)
           },
    effects:  [(lights_on,1.0)]
    ].
```

6. Harry: I have switched on Redlight.

7. Bob: Ok.

The utterances from (3) – (7) are then repeated for Bluelight and Greenlight. Just before the agents discharge their joint commitment (right after Harry informs Bob that he has turned on the Greenlight but Bob hasn't confirmed it yet), we interfere with the joint action by manually switching off Redlight that had previously been switched on by the agents. Bob notices the change and the following conversation ensues that establishes mutual belief about the discovered problem.

(18) Bob: Redlight is not switched on.

(19) Harry: Ok.

The agents recover from the unexpected problem, continue working on their joint action, and finally discharge their joint commitment. The joint commitment is discharged when all the lights are turned on and Bob established mutual belief about the final test action.

(23) Bob: All lights are switched on.

(24) Harry: Ok.

One can see that the above dialogue follows directly from joint intention theory. Utterance (1) is a REQUEST from Bob to Harry to jointly execute an action expression. Utterance (2) is an AGREE from Harry to Bob. At this point each agent has made its PWAG public and they are thus bound together by a joint commitment to have done the action "jointly turn lights on". Recall that the expression `(While p Do a)` is actually a shorthand for the action expression $(p?;a)^*|{\sim}p?$ which is an OR expression consisting of an indefinite repetition and a test action. As such, there is a sub-dialogue that the agents engage in to decide which component of the OR expression[6] to execute first. We have omitted this sub-dialog from the above dialogue transcript for clarity. The agents eventually decide to execute the indefinite repetition of the OR expression and Bob does its part of the action sequence in the indefinite repetition. The agents are following the lockstep policy for joint action execution so Bob INFORMs Harry via utterance (3) that its test action succeeded

[6]JI theory does not specify which action in an OR action expression to execute first. It assumes that the agents somehow end up choosing the right branch in the OR expression. We will see in the next chapter that one implementation strategy is to have the agents decide among themselves which action to execute first. It is important to note that this is just one possible approximation (or execution strategy) for joint OR action expression and this strategy is implemented in the current STAPLE interpreter.

because of its commitment to establishing mutual belief that it has done the test action. Our natural language generator translates `done(Bob,test(~switched_on(redlight)))` as "Redlight is not switched on". Thereafter, Harry turns on the Redlight, and informs Bob that it has turned on the Redlight (because of its commitment to establish mutual belief about this fact). Bob helps the mutual belief establishment by confirming the informed proposition (via "Ok") and the agents then move on to the next step in the action expression (finding another light that is not switched on, switching it on, and establishing mutual belief along the way).

Next, we explore how the agents behave when we modify the "observability conditions" in the world.

3.4.2 Modifying what the agents can see

We modify the above example as follows: (i) Both Bob and Harry know the initial state of the world. (ii) Both agents can only observe the effects of their own actions so that Bob cannot see what Harry did and vice-versa. (iii) The agents do not have a mutual belief that they can observe each other. (iv) Both agents are capable of performing action switch_on/1. (v) We modify the plan "jointly turn lights on" to consist of an action sequence in which the first action specifies that "Some Agent" turns on the `redlight`, after which Bob turns on the `bluelight`, which is followed by Harry turning on the `greenlight`. This action expression is specified in STAPLE by the sentence:

```
action(switch_on(redlight),SomeAgent),
action(switch_on(bluelight),bob),
action(switch_on(greenlight),harry)
```

"SomeAgent" is a capitalized atom, which indicates a variable in STAPLE (as in Prolog.) Bob, who establishes the joint commitment to execute the action expression, knows the actors for each action in the sequence. Harry does not know the identity of "Some Agent" and treats it as an unbound variable whose value is to be found later. However, Harry believes that Bob knows who will switch on the red light, that is, that there exists some K that Bob believes is the actor of the first action switch_on(redlight). This is specified in STAPLE via the following sentence.

```
bel(harry,exists(K,bel(bob,equals(K,i(Z,
                                actor(switch_on(redlight),Z)))))).
```

Here the functor '*i*' is the Russellian quantifier 'iota'. We define two more communicative acts to use with this example. First we define an INFORM-REF to be an inform of the referent of an expression [1], and then we declaratively define ASK-REF as a REQUEST to do an INFORM-REF. In STAPLE, it is specified as a plan (the STAPLE construct to define named action expressions) whose body consists of the following composed action:

```
request(X,Y, action(informref(Y,X,C,i(Z,P)),Y),Q)
```

The above action specifies a request from agent X to agent Y that agent Y inform agent X the referent C of the expression i(Z,P). The precondition of the ASK-REF communicative act is that the agent performing ASK-REF knows the referent it, that is, believes that there exists some K that the recipient of ASK-REF believes is the referent of the expression i(Z,P), that is, in STAPLE:

```
bel(self,exists(K,bel(Y,equals(K,i(Z,P)))))
```

The following actual dialogue between agents Bob and Harry continues.

Dialogue 2:

1. Bob: Let us jointly turn lights on.
2. Harry: I agree to jointly turn lights on.
3. Harry: Who will switch on Redlight?
4. Bob: I will tell you who will switch on Redlight.
5. Bob: You will switch on Redlight.
6. Harry: Ok, I will switch on Redlight.
7. <action: Harry pushes button; Redlight turns on>
8. Harry: I have switched on Redlight.
9. Bob: Ok.

As discussed earlier, the REQUEST from Bob to Harry in (1) and AGREE from Harry to Bob in (2) establishes the joint commitment between them for jointly executing the action expression. Utterance (8) onwards is similar to Section 3.4.1 where an agent who is specified to do an action as part of the joint commitment does that action and establishes

mutual belief that it has been done. Utterances (3)-(6) comprise a sub-dialogue to find out the actor of the first action in the joint action expression. This behavior contrasts with Breazeal's implementation [13] where the robot does the action if it can do it, and if not, looks towards the human for the human to do it[7].

In continuing to interpret the joint action expression, Harry discovers that it does not know the actor of the action `switch_on(redlight)` that it is committed to getting done. Therefore, it adopts an individual commitment for finding the agent who is the actor of action `switch_on(redlight)`. In STAPLE, this translates to Harry adopting a PGOAL to achieve the following proposition.

```
exists(K,bel(harry,equals(K, i(Z,actor(switch_on(redlight),Z)))))
```

Means-ends reasoning leads Harry to infer that it can achieve this committed goal by performing the ASK-REF communicative act[8]. It intends to perform ASK-REF, and then goes ahead and acts on its intention that results in utterance (3). Bob interprets (3) as a REQUEST for an action (according to the definition of ASK) and AGREES to that REQUEST by way of utterance (4). Utterance (4) can be considered superfluous though its effect is exactly what is predicted by the JI theory. The reasoning in STAPLE about communicative acts can be optimized using techniques similar to that used by Appelt [2] to enable Bob to figure out that it does not have to first AGREE to Harry's request for INFORM-REF rather, it can discharge Harry's PWAG resulting from utterance (3) by simply performing the requested INFORM-REF.

We have mentioned before that there are two widely used agent communication languages – KQML [68] and FIPA [41]. It has been pointed out in the literature [30, 66] that the semantics of these agent communication languages lacks several useful features. For example, the REQUEST communicative act in these languages does not commit the requester towards the requestee. Therefore, the requester is off the hook if he changes his mind (say, if he discovers something that makes his requested action impossible) after making the request. On the other hand, the semantics of REQUEST presented in the previous chapter ensures that the requester has a PWAG towards the requestee as soon as the REQUEST is done. As such, if the requester changes his mind after making the request, he will discharge his PWAG by establishing mutual belief with the requestee about this fact. So our semantics of REQUEST predicts that there will be communication

[7] Our animated characters can be customized to use pointing gestures thereby, enabling another modality for delivering communicative acts.

[8] There could be other acts as well that achieve this effect, such as perceptual ones.

if the requester changes his mind whereas this communication does not follow from the semantics of REQUEST in FIPA and KQML. The next example illustrates this specific scenario.

3.4.3 Oops! Never Mind

We repeat the example in Section 3.4.2 with the exception that (i) the sentence `bel(self, dark(outside))` is asserted into Bob's belief base before starting the experiment, and (ii) Bob establishes joint commitment with Harry relative to its belief that it is dark outside. We pause both agents (using pause buttons on each agent's console) right after Harry switches on the red light. Thereafter, we retract `bel(self,dark(outside))` from Bob's belief base and assert `bel(self,`$\sim$`dark(outside))` using the Prolog interface that is exposed by each STAPLE agent through its agent console. On resuming both agents, Bob discovers that its reason for establishing the joint commitment is no longer true, that is, its joint commitment is irrelevant (in JI terminology, the relativizing condition Q is false). This discovery is achieved by means of triggers that the STAPLE interpreter sets on the agent's belief base to monitor the discharge conditions of PGOAL, INTEND, and PWAG. The interpretation of Bob's PWAG by the STAPLE interpreter leads to Bob's adopting an individual commitment (PGOAL) to achieve mutual belief with Harry that the relativizing condition of its PWAG towards Harry is false. The following actual dialogue is observed[9].

(15) Bob: It is not dark outside.

(16) Harry: Ok.

This mutual belief discharges the commitment of each agent towards each other as per the definition of PWAG, and hence discharges the joint commitment between Bob and Harry. The main point here is that first principles reasoning achieved this effect only because the semantics of Bob's request to Harry to switch on the lights resulted in the right commitments as per JI theory. As discussed earlier, if the agents had been reasoning using FIPA's semantics of REQUEST, then this communicative behavior would not have been possible without additional inference rules. Next, we summarize the main points of the chapter.

[9]Recall that at present, agents exchange speech act representations. The natural language generator translates speech act formulae for the human.

3.5 SUMMARY

This chapter demonstrates the feasibility of using a logic-based declarative language to obtain team and communicative behavior automatically without having to program this behavior explicitly. The examples in Section 3.4 were created by merely encoding the initial conditions and stipulating the joint plan, from which team and communicative behavior followed automatically. The STAPLE agents exhibited team-oriented dialogue by interpreting the constructs of joint intention theory along with first principles reasoning over a formal semantics of communicative acts based on that theory. This research shows that formal semantics of communicative acts can be fruitfully employed for inter-agent dialogue.

STAPLE supports several other interesting team and dialogue behavior not discussed here, including dialogue in teams of more than two agents. Future work includes using gesture and facial expression to deliver a communicative act, plan recognition for full-fledged dialogue, and expression of turn-taking constraints as a joint execution strategy (akin to the lockstep one described in [70]). To conclude, the examples presented show that it is possible to obtain team-oriented dialogue by using formal semantics of communicative acts and joint intention theory without having to program that behavior explicitly.

Next, we discuss the implementation of the STAPLE interpreter employed for the examples illustrated in this chapter.

Chapter 4

Implementation of a STAPLE Interpreter

An operational semantics of STAPLE is presented in the appendix. This chapter presents an implementation of a STAPLE interpreter according to that operational semantics. We have seen that STAPLE constructs include a subset of modal logic, dynamic logic of actions, and temporal logic, as well as abstractions from joint intention theory and its extensions. Beliefs, goals, commitments, and intentions are represented explicitly in STAPLE, actions are required to have a logical representation that can be used for reasoning, and plans as well as conversation protocols are treated as complex action expressions consisting of action sequences, non-deterministic OR, concurrent actions, repetitions, and test actions. These STAPLE constructs have the same model theoretic semantics as that of the underlying logic from where they are borrowed and the STAPLE interpreter attempts to faithfully execute the logic directly.

The axioms of rational behavior are specified as STAPLE rules, and agent programmers can either provide new rules or override existing ones. We will see that these rules cannot affect the semantics of the basic constructs from JI theory (such as commitments and intentions) because of the way the basic theoretical constructs are interpreted[1]. STAPLE agents can have multiple simultaneous commitments and intentions, and a notion of importance is used to order everything from commitments and intentions to plans and rules. The syntax of STAPLE is presently an extension of the usual Prolog syntax with the exception that certain constructs such as primitive actions and plans can be written in both Prolog and Java.

The remainder of this chapter is organized in the following manner. An overview of the STAPLE interpreter is presented in the next section. The interpretation of individual

[1]For example, there is no way to modify the behavior of a PGOAL solely by modifying the declarative rules of rational behavior. One would have to modify the PGOAL interpreter (discussed later in this chapter) and recompile the STAPLE interpreter in order to modify the implemented semantics of PGOAL.

commitments and intentions is discussed in Section 4.2, and that of joint commitment between two agents, including the communication between them, is discussed in Section 4.3. Finally, we conclude in Section 4.4 with a summary of this chapter.

4.1 THE STAPLE INTERPRETER

The STAPLE interpreter needs logical reasoning capabilities along with the ability to handle procedural tasks such as control flow and stack manipulation. Logic programming languages such as Prolog are good for logical reasoning but procedural tasks can quickly get quite complex and unwieldy in such languages. It was quite clear from our experience [62] with early versions of STAPLE that completely implementing the STAPLE interpreter in Prolog was not a viable option because a large portion of the interpreter dealt with procedural control, and the commercially available multi-threaded Prolog implementations were too buggy and quickly broke down when used with STAPLE. The support for multi-threading was needed in the interpreter development platform to enable concurrent actions, asynchronous update of the belief base via multiple sensors (or observers in STAPLE terminology), and other functionality such as pausing and resuming an agent, asynchronously interacting with its belief base, and providing a responsive user interface. Similarly, imperative languages such as Java are good for procedural tasks but are ill-suited for logical reasoning. As such, we take a hybrid approach for the current STAPLE development by choosing to use both Prolog and Java, and using each language for tasks that they do best – Java is used for procedural control and Prolog is used for logical reasoning. We implemented a multi-threaded Prolog interpreter[2] in Java, thereby closely integrating these two disparate languages for use in the development of the STAPLE interpreter. Here, we present an overview of the STAPLE interpreter and its main components irrespective of their implementation language.

4.1.1 Overview

As in any interpreted language, a STAPLE program is first parsed and the appropriate data-structures are initialized. The agent's beliefs, actions, plans, and rules are placed

[2] A recent enhancement to the STAPLE interpreter by Natural Language Interaction LLC replaced this multi-threaded Prolog interpreter with single threaded Sicstus Prolog by embedding Sicstus within the STAPLE interpreter. As a result, the calls to the Prolog subsystem got serialized but the sheer speed of Sicstus has resulted in a two orders of magnitude overall speed-up in STAPLE programs and more speed-ups are expected upon further optimizations. Such embedding was not possible when STAPLE interpreter was being implemented for this dissertation.

in the appropriate databases (described next), its commitments and intentions initialize the stacks that are used to interpret and keep track of the progress of commitments and intentions, and any observers (abstraction of sensors), and actuators (abstraction of effectors) required by the agent at startup are activated. Thereafter, the main interpreter loop is started and execution of the agent proceeds so as to achieve its commitments and intentions. Figure 4.1 shows the main components of the STAPLE interpreter that make this execution of STAPLE agents possible.

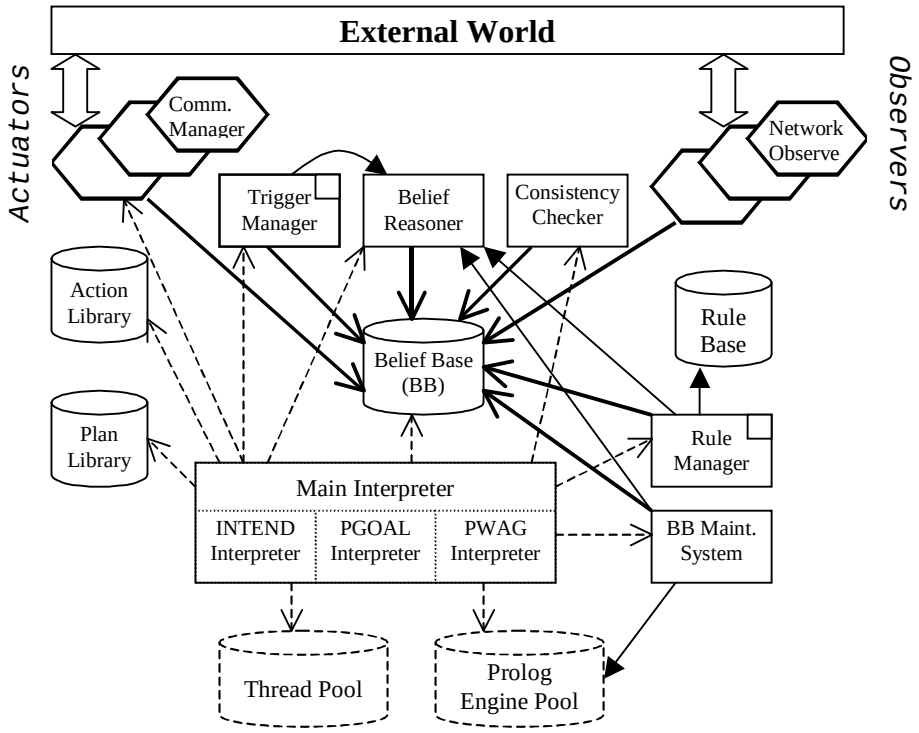


Figure 4.1: Main Components of STAPLE Interpreter

The directed arrows in Figure 4.1 show the dependency of the components – they indicate which components invoke or use which other components. The type of the arrows just indicates visual grouping of components. For instance, the bold arrows show the components that depend on the belief base, and the dotted arrows show the components that the main as well as the various modal interpreters depend on for their functioning. The components shown with notched corners (trigger manager and rule manager) have their own dedicated Prolog engines, and all other components that need access to a Prolog engine get it from a pool of reusable Prolog engines. All Prolog engines share the same thread-safe knowledge base that has a very fine-grained locking granularity to allow highly

concurrent access [60]. The belief reasoner and the belief base maintenance systems are written entirely in Prolog, the consistency checker and the rule manager are written partly in Java and partly in Prolog, and everything else is written entirely in Java. The main interpreter, and the INTEND and PGOAL interpreters are discussed in Section 4.2, the PWAG interpreter is discussed in Section 4.3, and all other components shown in Figure 4.1 are discussed next.

4.1.2 Action and Plan Library

All primitive actions accessible to an agent are instantiated using a data-structure that is treated as a Prolog term by the Prolog subsystem for the purpose of unification and also has an executable component. These actions include the actions common to all agents that are defined in the STAPLE library as well as the actions specific to that agent. Further, as mentioned earlier, these actions may have been written in either Prolog or Java. The action library keeps track of the action instances using the name of the action and its arity (i.e., the number of arguments that the action requires). This method of indexing actions may be viewed as a limitation in the current STAPLE implementation – no two actions can have exactly the same name and number of arguments[3]. The action library provides several ways to query for actions that meet a certain requirement. For instance, one can ask the action library for an action with a given name and arity, or for a list of actions whose intended or desired effect unifies with a given proposition. The action library returns a copy of the action instances that satisfy a given query. Each action instance may be queried for its attributes such as effects, precondition, and context among others. Therefore, the query for actions may also include additional criterion such as restricting the query to include only those actions whose precondition is satisfied in addition to the main criterion. Though not indicated in Figure 4.1, such additional criterion requires invoking the belief reasoner using a Prolog engine over the agent's belief base. Hence, the action library can be thought of as a "smart database".

The plan library is similar to the action library except that the data structure used to instantiate plans represents a complex action expression that is not executable (the constituent actions of a plan, however, may be executable primitive actions). The action expressions are treated as Prolog terms by the Prolog subsystem for unification purposes. The plan library can be queried in the same manner as the action library, and it returns

[3]However, it is not a serious limitation – actions can always be renamed to avoid any name conflicts that are reported as errors when the agent program is loaded.

copies of plans that match the query criteria.

4.1.3 Belief Base and Belief Base Maintenance System

The beliefs of a STAPLE agent are stored in a blackboard data structure that is treated by our Prolog engines as the "Prolog knowledge base". These beliefs include both the beliefs common to all STAPLE agents as well as those specified in the agent specification. Multiple Prolog engines, each running in a separate thread of execution, may access the same belief base concurrently. As such the belief base (BB) is designed to be thread-safe and has a fine-grained locking granularity (at the level of predicates with same arity). It is possible for one Prolog engine to update the belief base while another Prolog engine is executing the belief reasoner. As such, our Prolog engines have an execution semantics that is reasonable in multi-agent systems in the face of concurrent updates. For instance, it is possible that the belief reasoner decides that (BEL x p) is false even though (BEL x p) is asserted into the belief base while the belief reasoner is being executed. This may happen when (BEL x p) is asserted *just after* the Prolog engine executing the belief reasoner has read the possible candidates for SLD resolution from the BB. However, subsequent tests for (BEL x p) will return the correct expected answer. One can rationalize this behavior as the agent not being momentarily "aware" of a fact in its belief base during concurrent updates[4]. Similarly, each belief read from the BB is checked, *just before* using it for the SLD resolution, in a thread-safe manner to make sure that it has not been deleted from the BB after it was read. The belief base is usually updated through the BB maintenance system instead of asserting beliefs directly to it.

A belief base maintenance system complements the belief reasoner and is primarily needed to help it avoid circular loops and infinite recursions by making sure that all information asserted to the knowledge base is of a certain form. In particular, using the positive introspection property of beliefs in our logic, it reduces all facts of the form (BEL x (BEL x...(BEL x p)...)) to the equivalent fact (BEL x p) before asserting them in the knowledge base. Similarly, facts like (BEL x $\neg\neg\ldots\neg p$) are reduced using the equivalence (BEL x $\neg\neg p$) $\equiv$ (BEL x p). It also combines beliefs into their reduced form, for instance, if the belief base already has (BMB x y p) then when asserting (BMB y x p), the BB maintenance system retracts (BMB x y p) and asserts (MB x y p) using the equivalence

[4]It is possible to imagine alternate execution semantics where a Prolog engine restarts its reasoning process whenever the belief base is updated by another Prolog engine. However, there are several issues with such a semantics that are not very clear. The current execution semantics is cleaner and easier to implement.

(BMB x y p) $\wedge$ (BMB y x p) $\equiv$ (MB x y p) in our logic. The belief reasoner can deduce both the original facts from the asserted fact. We use the convention that an agent believes everything in its belief base. So p and (BEL self p) in the BB mean the same thing to the belief reasoner. However, the BB maintenance system asserts p instead of (BEL self p) to enable the Prolog engine to take advantage of first argument indexing. Similarly, the BB maintenance system asserts (MB self y p) irrespective of whether (MB self y p) or (MB y self p) is being asserted using the equality (MB x y p) $\equiv$ (MB y x p). This convention of always keeping "self" as the first argument of mutual belief terms simplifies the belief reasoner and makes it more efficient as there need not be any deduction rules for the equality (MB self y p) $\equiv$ (MB y self p). This is particularly so because most queries about mutual beliefs inquire whether or not there is mutual belief between this agent and another agent about some proposition. A belief base maintenance system should also keep the BB in a consistent state at all times. However, we have yet to implement a comprehensive consistency maintenance capability in our BB maintenance system. This is known to be a hard problem and we do not address this problem at present other than checking for contradictory terms such as p and $\neg p$.

4.1.4 Belief Reasoner and Consistency Checker

The language being interpreted uses a subset of modal logic, temporal logic and dynamic logic of actions so an agent's belief base consists of statements in these logics. We need a belief reasoner equipped with the axioms and inference rules from these logics to be able to prove whether or not a formula is true by reasoning over an agent's belief base. It is important to note that this belief reasoner *does not infer all possible formulae* that follow from the belief base, rather, its task is the following: Given a proposition p, it decides whether or not p follows from the belief base using the provided inference rules. Table 4.1 lists a subset of the inference rules that the modal reasoner currently uses. These rules are shown for right to left provability – for instance, prove (bel α p) in order to prove (bel α (bel α p)).

As mentioned earlier, we use the convention that an agent believes everything in its belief base. So p and (bel self p) in the BB mean the same thing to the belief reasoner. Beliefs about other agents are represented just like any other fact in the BB. For instance, "I believe that x believes p" will be asserted into the agent's BB as (bel x p) that is a simplified form of (bel self (bel x p)). Certain extra-logical checks[5] as well as support from

[5]For instance, the rule (bel α p):- (bel α $\Box p$) is used only if p is not of the form $\Box q$ where q is some

Table 4.1: Sample Deduction Rules for Belief Reasoner

(BEL self p) :- p	(BEL α $\neg\neg$p) :- (BEL α p)
(BEL α p) :- (BEL α $\Box$p)	(BEL α $\Box\Box$p) :- (BEL α $\Box$p)
(BEL α $\Diamond$p) :- (BEL α p)	(BEL α $\Diamond\Diamond$p) :- (BEL α $\Diamond$p)
(BEL α (BEL α p)) :- (BEL α p)	(BEL α p) :- (BEL α q =>p) $\wedge$ (BEL α q)
$\neg$(BEL α p) :- (BEL α $\neg$p)	(BEL α p$\wedge$q) :- (BEL α p) $\wedge$ (BEL α q)
$\neg$(BEL α p) :- \+ (BEL α p)	(BEL α $\Box\neg$(p$\wedge$q)) :- (BEL α $\Box\neg$p) $\vee$ (BEL α $\Box\neg$q)

the belief base maintenance system are used to avoid circular and infinite reasoning by the modal reasoner.

The consistency checker uses the belief reasoner to ensure that an agent does not adopt any commitment or intention that conflicts with any existing commitment or intention of that agent. For instance, the agent cannot adopt an intention to achieve $\neg p$ if it already has an intention to achieve p (ideally speaking, it can have the intention to achieve $\neg p$ and p at different times[6]). An agent cannot adopt an intention or commitment to achieve p if it believes that the new commitment or intention makes an already existing commitment or intention impossible, that is, if (BEL x p) $\supset$ (BEL x $\Box\neg q$) where the agent has an existing commitment or intention to achieve q. Similarly, an agent cannot adopt an intention or commitment to achieve p if it believes that the new commitment or intention is rendered impossible by an existing commitment or intention. An intention to perform an action a (or a commitment towards being in a state where that action has been done) is inconsistent with an intention or commitment to achieve a proposition p if at least one of the following holds: (1) the proposition p is contradictory with either the effect, or the precondition, or the context of the action[7], (2) the proposition p makes either the precondition or the context of the action impossible, or (3) the effect of the action makes achieving the proposition p impossible, that is, effect(a) $\supset \Box\neg p$. Similarly, an intention or commitment for performing an action is inconsistent with the intention or commitment for performing another action if one of the following holds: (1) the effect of

proposition. Similarly, checks for logical variables (Prolog var/nonvar) are used where needed.

[6]The current implementation of consistency checker ignores this timing subtlety. Future versions of STAPLE will include a check for timing inconsistencies to a certain extent using predicates to specify timing constraints on achievement of commitment propositions.

[7]Some of these inconsistencies may be resolved by reordering the timings, for example, the start time of actions that need to be done or timing of when the agent works towards achieving a commitment. STAPLE does not currently support ordering constraints between independent commitments and intentions and so is unable to resolve such inconsistencies.

one of the actions is contradictory with either the effect[8], or precondition, or the context of the other action, or (2) the effect of one of the actions makes either the precondition or the context of the other action impossible. An intention or a commitment for performing either a sequence, or a concurrent action expression, or a repetition of singleton actions is inconsistent with another commitment or intention if either (1) the intention or commitment for even one action in the action expression is inconsistent with the other commitment or intention as per the above discussion, or (2) the intention or commitment for the entire action expression treated as a single action (such as the action expression of a plan wherein the entire action expression may have an effect, a precondition, and a context) is inconsistent with the other commitment or intention. Similarly, an intention or a commitment for performing a non-deterministic OR of singleton actions is inconsistent with another commitment or intention if either (1) the intention or commitment for every action in the action expression is inconsistent with the other commitment or intention as per the discussion earlier, or (2) the intention or commitment for the entire action expression treated as a single action is inconsistent with the other commitment or intention. The inconsistency of an intention or a commitment for performing a complex action expression involving any combination of the action formation operators is similarly determined by recursively decomposing the action expression, and using appropriate tests as discussed above.

4.1.5 Trigger Manager, Rule Base, and Rule Manager

The trigger manager supports triggering on propositions (setting, checking, firing, and removing triggers) in the agent's belief base. The trigger manager maintains an internal database of triggers. The triggers are not checked on every update of the belief base but are checked only when the trigger manager is explicitly asked to check triggers (usually by the main interpreter). Every trigger has an associated action that is executed when the trigger fires. The trigger action typically informs the trigger setter about the firing of the trigger so that it can take the appropriate action. Triggers also include some convenient fields that allow the trigger setter to quickly determine the conditions that became true resulting in the firing of that trigger instead of having to invoke the belief reasoner all over

[8]This is a simplification due to ignoring the timing subtlety as mentioned earlier. For example, the present STAPLE interpreter will not allow an agent to have an intention to open the door and another intention to close the door at the same time (although opening the door and then closing it can be part of an action sequence because the agent will have the intentions for these individual actions at different times as per the semantics of intending action sequences).

again.

We distinguish between the declarative rules interpreted by the rule interpreter and the inference rules[9] used by the main as well as the modal (INTEND, PGOAL, and PWAG) interpreters. The rules interpreted by the rule interpreter tell what to do when certain facts can be inferred from the agent's belief base. For example, the rule to intend the highest utility action or plan that can achieve a committed goal is one such rule. An agent programmer can provide the rules that are interpreted by the rule interpreter and can even override the default rules provided by the STAPLE library. The importance associated with each rule is used to select one applicable rule when multiple rules are applicable to a situation. The STAPLE default rules (the rules in the STAPLE library that are accessible to all agents) can be overridden by providing rules with higher importance. The antecedents and the consequents are in conjunctive normal form. The rule interpreter can be used (1) to repeatedly select rules applicable to the current state of the knowledge base (rules whose precondition is the constant "true") and fire the selected rules until no more applicable rules are found, and (2) to select rules applicable under a specified precondition, and fire the most important rule among those selected. The rule interpreter uses a dedicated Prolog engine for its use (such as finding applicable rules, testing antecedents, and executing the consequents) and it invokes the belief reasoner to evaluate the applicability criterion as well as the other antecedents of a rule. At present, the STAPLE rule base is stored in a Prolog knowledge base in a similar fashion as the belief base. The rules are fairly static but new rules may be inserted and existing ones may be removed from the rule base at run time.

4.1.6 Observers and Actuators

Observers are high-level abstractions for interfaces that monitor the external world via sensors, sockets etc. It is desirable that observers be active components executing concurrently with the main interpreter because they monitor the world in real time. Otherwise, the main interpreter will have to poll all the observers for relevant information during every cycle and execute their associated code. Delegating these procedural tasks to the observers

[9]The declarative rules interpreted by the rule interpreter are accessible to the agent programmer. However, the inference rules used by modal interpreters are built into the interpreter itself (i.e., hard-coded). They implement the semantics of the modal constructs as per the theory and they cannot be modified by the agent programmer. Further, the inference rules used by the main interpreter (such as when to perform intention reconsideration) are currently built into the main interpreter but it can be made declarative and accessible to the agent programmer in future implementations of the STAPLE interpreter.

also allows the main interpreter to be dedicated to logical interpretation. Therefore, observers in STAPLE are active components, each observer being executed in a separate thread. A typical observer receives data from the sensors it is monitoring in real time, extracts the relevant information, and asserts the new information in the agent's belief base as a logical formula that the STAPLE interpreter can reason about. An observer can also be used for *reflex actions* such as automatic acknowledgement upon receipt of a message. It can also be used for *peripheral processing*, for example, when an observer monitoring a TCP socket connection receives an INFORM, it need not assert only this fact but also the inferred goal and intention of the sender (using the speech act definition of INFORM). An observer that performs logical inference as part of peripheral processing may need to use a Prolog engine from the common pool of Prolog engines. Figure 4.1 represents observers as hexagons, and shows the network observer that is common to all STAPLE agents. The network observer performs three tasks – it listens on a server socket (the advertised network address of the agent) for connection requests, reads messages from existing network connections (and asserts a logical representation of the messages into the agent's belief base), and watches for existing connections that close abnormally (in which case, it asserts that information into the belief base). The STAPLE interpreter starts the observers during agent start-up.

Actuators are also high-level abstractions that are used to control and monitor physical effectors. They may run in separate threads, and possibly perform physical actions that may extend over time. Actuators are typically invoked from the "code" of an action when that action is executed. An actuator may report on the progress, completion, and success of its task by asserting appropriate facts into the agent's belief base. Figure 4.1 represents actuator as hexagons, and shows the communication manager that is an actuator common to all STAPLE agents. The communication manager can be tasked to send messages to other agents, and to connect to those agents if not already connected (in which case, it will inform the network observer to add the new connection to its list of connections being observed).

4.1.7 Thread Pool and Prolog Engine Pool

The STAPLE interpreter includes shared pools of reusable execution threads and reusable Prolog engines. As such, most components of the STAPLE interpreter that need to access the Prolog engine only occasionally can use engines from the shared pool when needed. The threads from the thread pool are used: (i) by the main interpreter to execute actions

concurrently, (ii) by active components such as the observers and actuators that run asynchronously, and (iii) by the user interface components such as the Prolog console exposed by each agent. Actions whose code is in Prolog are executed by the main interpreter in a thread from the thread pool using a Prolog engine from the engine pool. We will see in the next section that the main interpreter concurrently executes the most important actions of the agent.

We now present the main STAPLE interpreter along with the modal interpreters for executing individual commitments (PGOAL) and intentions (INTEND). These interpreters make use of the components discussed above for their functioning.

4.2 EXECUTING INDIVIDUAL COMMITMENTS AND INTENTIONS

The heart of the STAPLE interpreter for executing single agents consists of the main interpreter and the modal interpreters for executing individual commitment (PGOAL) and individual intention (INTEND). The main interpreter corresponds to the central interpreter in traditional agent language interpreters and agent architectures. It runs in an infinite loop driving the agent towards achieving its goals and discharging its commitments and intentions. It also invokes the modal interpreters for PGOAL and INTEND to interpret these terms as per their definition in our logic. These interpreters use a stack-based data structure to keep track of progress, to interpret action expressions, and to follow the definitions of the modal terms. STAPLE agents can have multiple simultaneous commitments and intentions. Each high level commitment and intention is assigned a separate stack. The built-in actions adopt and subgoal can be used from within rules as well as from other actions and plans to manipulate those stacks. The adopt action refers to the creation of a new high-level commitment or intention by creating a new stack and pushing the adopted term on it. The subgoal action refers to the creation of a sub-commitment or sub-intention with respect to another commitment or intention by pushing the subgoaled term on the stack that had the higher-level commitment or intention on top of it. In the remainder of the dissertation, we will use the term “intend stack” or “intention stack” to indicate that a stack has an INTEND term on top of it, and use the term “commitment stack” to indicate all other stacks. We now first present the main interpreter and then discuss the modal interpreters for executing PGOAL and INTEND.

4.2.1 Main Interpreter

The main interpreter uses a simple mechanism based on utility and penalty to incorporate multiple simultaneous commitments and intentions. A rational agent will give more importance to a commitment or intention having higher utility. Such an agent will also give more importance to a commitment with higher penalty [99]. We assume an importance function that combines the utility and the penalty of a commitment to compute its overall importance to the agent. As mentioned earlier, we syntactically augment PGOAL and INTEND to specify their importance as an extra argument. Importance can be a function in general but a numeric value (constant function) will suffice for now. The present version of STAPLE does not allow utility and penalty to be specified separately[10]. We give a meaning to the notion of importance by relating it to the axiom of rational behavior that is used by the main interpreter to act on intentions. This meaning[11] is consistent with the definitions of PGOAL and INTEND in Chapter 2 and is specified as Axiom 4.1.

Axiom 4.1. *Axiom of Rational Behavior: A rational agent will act on its most important executable intentions at all times.*

Let Intend(t) = {Intend_1, Intend_2,..., Intend_N} be the set of executable intentions of an agent at time t and let $\text{i}_k \; \varepsilon \; \mathbb{I}$ be the importance function of Intend_k where $1 \leqslant \text{k} \leqslant \text{N}$, and $\mathbb{I}$ is the set of importance functions. Also, let eval: $\mathbb{I} \rightarrow \mathbb{R}$, where $\mathbb{R}$ is the set of real numbers. The set of most important actions at time t for this agent is given by the intended actions in the set rational(t), where

$$\text{rational(t)} \equiv \{\text{Intend}_m \in \text{Intend(t)} \mid \text{eval}(\text{i}_m) = \max\,[\text{eval}(\text{i}_k)]_{1 \leqslant k \leqslant N}\}$$

Given this axiom, in each cycle the interpreter executes only the most important intentions and switches among intentions as their importance changes. Also, note that multiple intentions can simultaneously be most important and hence they will be executed concurrently (assuming that these actions do not interfere[12] with each other). In fact, this is how concurrent actions get executed at the same time. The definitions of PGOAL and

[10] However, the STAPLE interpreter may support separate specifications of both utility and penalty at a later time.

[11] One can think of various other methods to incorporate the notion of importance into the logic. For instance, the importance condition may be included as a conjunct in the relativizing condition of the PGOAL.

[12] It is the agent programmer's responsibility to specify the precondition, context, and effect of actions in such a way that two actions that interfere with each other will be found to be inconsistent by the STAPLE interpreter. Therefore, an agent will never have two intentions at the same time for doing these conflicting actions as discussed earlier. Hence, the assumption that actions executed simultaneously will not interfere with each other is valid.

INTEND do not preclude an agent from having multiple simultaneous PGOALs and INTENDs that are consistent with each other. The notion of importance as used above will be consistent with the modal logic definitions of PGOAL and INTEND if (1) the computation of importance does not have the side effect of making inconsistent the PGOALs and INTENDs of the agent that were previously consistent with each other, and (2) the axiom of rational behavior does not make any PGOAL or INTEND of the agent impossible. The first condition is trivially satisfied if the importance is specified as a constant function that returns a numeric value (as in the current version of STAPLE). In all other cases, it is the agent programmer's responsibility to ensure that this constraint is satisfied. The second condition essentially relates to timing constraints. For instance, STAPLE's executing the most important intentions at time t may make an intention with a lower importance impossible if that intention is for performing an action at time t. This problematic condition may be taken care of in the STAPLE interpreter by suitably increasing the importance of time critical commitments and intentions as their deadline approaches. Our implementing such an algorithm would require supporting predicates such as "at" and "by" to specify timing constraints on actions[13]. However, the second condition is trivially satisfied at present because STAPLE does not currently support actions with timing constraints. The main interpreter implements the above axiom of rational behavior.

Table 4.2 shows the main interpreter loop after the agent program has been initialized. Compared to the standard agent control loops (Wooldridge [118]), conspicuously absent from this figure are the steps to fetch a percept, to do belief revision depending on the percept, to get the next action of a plan for execution, and to test for soundness of the plan. The step of fetching the next percept is absent because the observers in a STAPLE program run continuously in a separate execution thread and update the belief base asynchronously. Belief revision is performed whenever the belief base is updated through the BB maintenance system. The belief base is usually updated when the observers receive new information, and also when an action has been successfully executed (in which case the effect of that action is asserted). The step of getting the next action to be executed is absent because the modal interpreters for PGOAL and INTEND initialize each stack appropriately for the next action to be executed for that stack, and multiple actions (those associated with the agent's most important intentions) are executed concurrently. The execution of multiple simultaneous actions contrasts with other BDI agent infrastructures such as IRMA [12] that execute only one action at a time. The IRMA architecture is

[13] As noted earlier, this feature is one of the future enhancements of the STAPLE interpreter.

discussed further in Chapter 9. A STAPLE plan is applicable when the context of the plan is true and the plan has not been achieved, and it is not impossible or irrelevant. The applicability of plans is checked not in every cycle of the main interpreter but via triggers on the belief base. In each cycle, the main interpreter checks to see if any triggers have fired. Firing of triggers is one of the conditions for intention reconsideration in a STAPLE agent.

Lines 2 and 3 in Table 4.2 implement intention reconsideration in the STAPLE interpreter. First, all applicable non-specific rules (i.e., rules with precondition "true") are fired repeatedly until no more rules are applicable. This may result in several new commitments and intentions being adopted. Thereafter, the modal reasoner for PGOAL is invoked to interpret the PGOAL stacks, and to fire any rules that may lead to the agent adopting or subgoaling intentions to achieve the various commitments. Lines 5-18 interpret and execute intentions repeatedly in a loop until one of the conditions for intention reconsideration becomes true. Note that breaking out of this inner loop eventually leads to intention reconsideration at the beginning of the next cycle of the outer infinite loop. As mentioned earlier, the firing of triggers is one of the conditions for intention reconsideration (line 15). Intention reconsideration is also performed when there are no more executable intentions. This is the case when either there is no intention stack (line 5 and 7), or when the status is "waiting" for all the intention stacks (line 15), that is, the terms on top of all stacks have set triggers on the belief base and are waiting for some proposition to become true, false, or impossible. There is also an explicit reconsider function as an OR condition in line 15 that returns false by default but that may be overridden by agent programmers to provide any additional criterion for intention reconsideration. Line 6 invokes the modal interpreter for INTEND to interpret all intend stacks as per the logical definition of INTEND, and to interpret action expressions. The main interpreter executes the most important intentions in line 11 as per Axiom 4.1. Here, all those actions are executed concurrently in a separate thread from the thread pool, and the main interpreter waits for all actions to finish executing before continuing further. Thereafter the trigger manager is invoked on line 13 to check if any triggers have fired. In our logic, (DONE a) means that the action a has just been done. So a natural question is when should an action that was "just done" be not considered as "just done" but as something that was "done earlier"? This transition happens in line 14 where all terms of the form (DONE a) in the belief base are replaced by (EARLIER (DONE a)). Lines 20-25 invokes the modal reasoner for PGOAL and INTEND to take any necessary actions due to firing of

Table 4.2: Main Interpreter Loop

```
1:  WHILE (TRUE) DO
2:      fireApplicablerules();
3:      interpretPGOALs();
4:
5:      WHILE(there exists at least one intention stack) DO
6:        interpretIntentions();
7:        IF (there is no intention stack) THEN
8:                break from inner while loop;
9:        END-IF
10:
11:        executeMostImportantIntentions();
12:
13:        checkTriggers();
14:        moveDoneToEarlier();
15:        IF (triggerFired || all intend stacks waiting ||
              reconsider())
           THEN
16:            break from inner while loop;
17:        END-IF
18:    END-WHILE
19:
20:    IF (triggerFired) THEN
21:        interpretPGOALs();
22:        interpretIntentions();
23:        reset triggerFired to FALSE;
24:        continue to the next loop of the outer while loop;
25:    END-IF
26:
27:    IF (there is no stack || all stacks waiting) THEN
28:        wait to be notified of new information by observers;
29:    END-IF
30:  END-WHILE
```

triggers. In lines 27-29, the main interpreter waits for new information to arrive because either there are no more stacks or the status of all stacks is waiting (i.e., the status is "waiting" for the terms on top of all stacks). In this case, there is nothing else to do so the main interpreter essentially goes to sleep and is woken up when an observer receives any new information. We now discuss the specifics of the modal interpreters for PGOAL and INTEND invoked by the main interpreter.

4.2.2 PGOAL Interpreter

The notion of an agent's commitment towards achieving some state in the world is expressed as a persistent goal or PGOAL, and the modal interpreter for PGOAL attempts to faithfully follow the formal definition of PGOAL. Recall that we assume that agents are competent with respect to their individual commitments, that is, if an agent believes that it has a PGOAL, then it does in fact have that PGOAL. Formally, we restate the assumption from Chapter 2:

Assumption 4.1.

$$\models (\text{BEL x } (\text{PGOAL x } p\ q)) \supset (\text{KNOW x } (\text{PGOAL x } p\ q)) \supset (\text{PGOAL x } p\ q)$$

We treat everything in an agent's belief base as being specified with respect to the agent's cognition. Therefore, a PGOAL specified in the agent specification file, or created by execution of a rule means that the agent believes that it has that PGOAL. Therefore, Assumption 4.1 enables the modal interpreter to interpret PGOAL as per Definition 2.1 by moving the PGOAL outside of the belief modality. A consequence of the above assumption is that an agent *knows* that it cannot drop its PGOAL at least until one of the conditions in the UNTIL clause becomes true. The conditions in the UNTIL clause are enforced using triggers and stack manipulation. They are checked before a PGOAL is pushed onto a stack, and then triggers are set on them for subsequent monitoring. One of the first things that the modal interpreter for PGOAL does is to check if any triggers have fired for commitments that already exist. When a trigger fires, the relevant stack is manipulated appropriately. For example, when a trigger fires due to a committed goal becoming impossible, then all stack items above the relevant PGOAL are discarded (after appropriate cleanup) and that PGOAL is re-evaluated. When a PGOAL is being adopted or subgoaled, the modal interpreter first checks for its achievement, impossibility, and irrelevance by executing the belief reasoner in a Prolog engine over the agent's belief base. For instance, to check for impossibility, the belief reasoner is invoked to check if $\Box\neg p$ can

be concluded from the belief base (this reasoning will make use of any domain dependent applicable rules of the form $\Box\neg p$:- q). Thereafter, the consistency checker is invoked to make sure that the PGOAL is consistent with the existing commitments and intentions of this agent, and triggers are set for any new commitments to monitor escape conditions in the definition of PGOAL as mentioned earlier.

A commitment to execute a complex action expression is interpreted by breaking it down into commitments for individual actions as per the rules in Table 4.3. These rules (reproduced here from Table 2.2 for convenience) are theorems in the logic and were discussed earlier in Section 2.3.1. They describe the encoded behavior of the PGOAL interpreter. These rules implement the semantics of PGOAL for executing action expression and therefore, they are not declaratively specified and are not modifiable by an agent programmer.

Each action for an agent is considered a separate event and is given a unique event identifier that distinguishes it from any other action ever intended or committed to by that agent, even if they are of the same event type. The event identifier includes a loop identifier that distinguishes the different instances of an action being performed in a repetitive loop as being separate events. The test action p? is treated as a single action that causes the belief reasoner to try to prove p, and this action may bind any logical variables in p in the process. The rule for action sequences in Table 4.3 essentially requires an agent to decompose the action sequence, committing to each of the actions one by one relative to the larger commitment. The action sequence is achieved if all the actions in the sequence are achieved – failure (or impossibility) of even one sub-action amounts to failure (or impossibility) of the entire action sequence. Two or more actions are concurrent if the agent has a commitment to do each action relative to the larger commitment, and when one of the actions is about to happen next, all the other actions are also about to happen next. The condition for achievement, failure, or impossibility of a concurrent action expression is the same as that of an action sequence. Non-deterministic OR is approximated by repeatedly selecting one of the actions at random out of the untried actions until either one action succeeds or all actions are found to be impossible. The commitment to wait for a proposition to become true, i.e PGOAL for the action *wait_for(p)*, is interpreted by first checking if p is true or false, failing which, triggers are set for p and $\neg p$ and the status of the PGOAL is set as "waiting"[14]. These rules translate into stack manipulation by the

[14]One possible enhancement to this scheme is to wait for a prespecified time and decide that p is impossible if that time elapses and the agent does not yet believe p or $\neg p$.

main modal interpreter similar to that illustrated for intention in Figure 4.2.

Table 4.3: Rules for Interpreting Action Expressions

Action Sequences

$\models$ (PGOAL x (DONE a_1;a_2;...;a_n) q)$\wedge\neg$(BEL x (EARLIER (DONE a_1)))
$\supset$ (PGOAL x (DONE a_1) (PGOAL x (DONE a_1;a_2;...;a_n) q))
$\models$ (PGOAL x (DONE a_1;a_2;...;a_n) q)$\wedge$ (BEL x (DONE a_1;a_2;...;a_i))
$\supset$ (PGOAL x (DONE a_{i+1};...;a_n) (PGOAL x (DONE a_1;a_2;...;a_n) q))

Repetition

$\models$ (PGOAL x (DONE a*) q) $\supset$ (PGOAL x (DONE a;(a)*) q)

Concurrent Actions

$\models$ (PGOAL x (DONE $a_1||a_2||\ldots||a_n$) q) $\supset$
(PGOAL x (DONE a_1) (PGOAL x (DONE $a_1||a_2||\ldots||a_n$) q)) $\wedge$
(PGOAL x (DONE a_2) (PGOAL x (DONE $a_1||a_2||\ldots||a_n$) q)) $\wedge$
$\vdots$
(PGOAL x (DONE a_n) (PGOAL x (DONE $a_1||a_2||\ldots||a_n$) q)) $\wedge$
[$\forall$k, 1$\leqslant$ k $\leqslant$ n (HAPPENS a_k) $\supset$ (HAPPENS a_1)$\wedge$...$\wedge$ (HAPPENS a_n))

Non-Deterministic OR

$\models$ (PGOAL x (DONE $a_1|a_2|\ldots|a_n)q$) $\supset$
(PGOAL x (DONE a_1) (PGOAL x (DONE $a_1|a_2|\ldots|a_n$) q)) $\vee$
(PGOAL x (DONE a_2) (PGOAL x (DONE $a_1|a_2|\ldots|a_n$) q)) $\vee$
$\vdots$
(PGOAL x (DONE a_n) (PGOAL x (DONE $a_1|a_2|\ldots|a_n$) q))

Finally, the rule interpreter is invoked once for each commitment in an attempt to find an intention that might achieve the committed goal. For instance, if it is believed that there are actions (including plans) that can achieve the committed goal, then a non-deterministic OR of all those actions is intended. Similarly, a belief that another agent can perform an action that can achieve a committed goal may result in an intention to request that agent to achieve it. If there are no more actions that can achieve a committed goal, and no agent is known that can achieve that goal, then the attempt to achieve that

goal is assumed to be have failed at that time. On subsequent invocation of the modal interpreter, a rule may reinstate the status of a failed commitment so that it can be tried again, or a rule may conclude that commitment to be impossible to achieve, say, if it has failed a given number of times. As per Definition 2.1 of PGOAL, an agent can drop a commitment that has been achieved or impossible or irrelevant, and this is done by removing such a commitment from the stack, and notifying the status of that commitment to its higher-level commitment or intention.

Next, we briefly discuss the modal interpreter for interpreting an agent's intentions.

4.2.3 Intend Interpreter

An intention to do an action a relative to q is represented by (INTEND x a q) and it is defined as a persistent goal in which the agent x is committed to performing the action a believing throughout that it is doing the action [25]. As such, the interpretation of INTEND is similar to that of PGOAL. Also, recall that the intending agent must be the actor of the intended action, and therefore, an agent cannot have an intention that some other agent does an action (though it can have commitment that the other agent does that action).

As in the case of PGOAL, the modal interpreter for INTEND first checks for fired triggers, and then re-evaluates any intentions that have either been achieved, or are believed to be impossible or irrelevant. Before pushing any new INTEND on the stack, the modal interpreter first checks to see if the intended action has been done, or is impossible or irrelevant, and then invokes consistency checker to ensure that the new intention is consistent with the existing commitments and intentions of this agent. An action is believed to be impossible if the context of the action is false, or if its precondition is false, of if the proposition $\Box\neg$(DONE self a) is true. As in the case of PGOAL, complex action expressions are believed to be impossible if any required action in the action expression is impossible – for instance, if any action in an action sequence, or concurrent action is impossible, or if all actions in a non-deterministic OR-expression are impossible. Also, triggers are set to monitor the escape conditions for any new intentions.

Intentions for complex action expressions are interpreted by marching through the action expression as in the case of PGOAL using rules similar to that in Table 4.3. These rules translate into stack manipulation by the main modal interpreter as illustrated in the examples in Figure 4.2. The agent in the Figure 4.2 has two simultaneous high level intentions for action expressions (p?;a)|b and ((c;d)||(e;f))* resulting in two independent

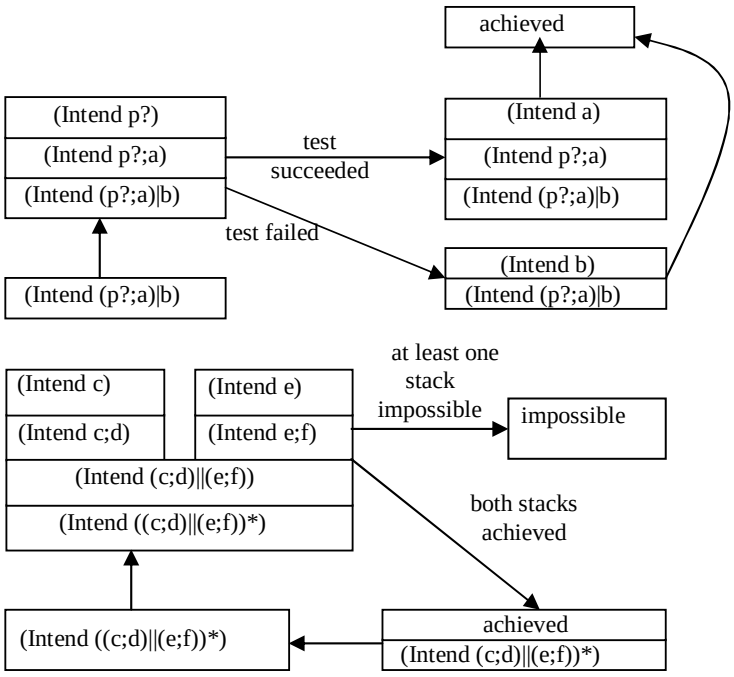


Figure 4.2: Rules for Interpreting Action Expressions

stacks. The arrows show how the stacks change with time. In the first example, we assume that the random selector for non-deterministic OR chooses (p?;a) to be executed first. The alternative action (b) is tried if (p?;a) fails. The second example starts with one item (Intend ((c;d)||(e;f))*) on the stack. The sub-stacks for concurrent actions (c;d) and (e;f) are treated as equally important independent stacks.

Next, we extend the single agent interpreter discussed so far to joint commitment by agent teams.

4.3 ESTABLISHING AND EXECUTING JOINT COMMITMENTS

Recall that the joint intention theory provides a formal model of teamwork – agents having a joint commitment to do an action are said to form a team to do that action. The joint commitment between two agents x and y to achieve a proposition p with respect to q is formally expressed as a joint persistent goal (JPG x y p q) between those agents (Definition 2.6). From [70], a joint commitment between two agents implies individual commitment by each agent. Therefore, a JPG between x and y for the proposition (DONE

y a) also commits agent x to agent y's doing the action a. Agent x can then act on its individual commitment (PGOAL x (DONE y a)) by being co-operative, helpful, or in any other suitable manner. For example, if the joint commitment is for agent y to move a chair out of the room, then agent x can act proactively to help agent y and open the door if the door is closed. The definition of JPG also requires establishment of mutual belief by both parties in the event of private belief regarding achievement, impossibility, or irrelevance of the jointly committed goal. A persistent weak achievement goal (PWAG) is the building block of joint commitment between two agents.

Also, recall that the persistent weak achievement goal of an agent x towards another agent y to achieve p relative to q, denoted by (PWAG x y p q), is another central concept in the teamwork theory and is used in the definition of the various communicative acts. A PWAG defines the one-way commitment of one agent towards another and represents a social commitment, provided that it is made public. We showed in Theorem 2.1 that mutual belief in each other's PWAG towards the other to achieve a goal p is sufficient to establish a joint commitment to achieve p provided that (1) there is mutual belief that p has not already been achieved, and (2) the PWAGs are interlocking, that is, one PWAG is relative to the other. Formally, we showed that

$$\models (\text{MB x y (PWAG x y } p\ q)) \wedge (\text{MB x y (PWAG y x } p\ r \wedge q)) \wedge (\text{MB x y } \neg p) \supset (\text{JPG x y } p\ r \wedge q), \qquad \text{where } r = (\text{PWAG x y } p\ q)$$

A consequence of this property is that it is possible to establish joint commitment between two agents by using communicative acts to establish mutual belief in each other's PWAG towards the other agent. As such, we have two choices for supporting joint commitment in STAPLE – we can either interpret JPG and JI directly, or we can instead chose to interpret PWAG. Recall that in any STAPLE program, we consider all explicitly represented terms to be with respect the agent's cognition. Therefore, the interpreter would need to implement whatever follows from (BEL x (JPG x y p q)) or (BEL x (PWAG x y p q)). Choosing to use (BEL x (JPG x y p q)) requires that we either make appropriate assumptions to move the JPG out of the belief modality so that the interpreter can just follow the definition of JPG, or we need to add suitable axioms to our logic that tells what to do when the agent concludes something from its belief about its joint committment with another agent. However, it turns out that we don't have to make any additional assumptions or changes to the logic if we use (BEL x (PWAG x y p q)). We showed in Lemma 2.1 that if an agent believes that it has a PWAG towards another agent, then it

does have that PWAG towards that agent. Formally,

$$\models (\text{BEL x } (\text{PWAG x y } p\ q)) \supset (\text{PWAG x y } p\ q)$$

This property follows from the definition of PWAG and the Assumption 2.1 that agents are competent with respect to their individual commitment, and it allows the modal interpreter to interpret PWAG by simply following the logical definition of PWAG (Definition 2.8). As such, teamwork is supported in the current[15] STAPLE implementation by the interpretation of PWAG. Also, the above property is explicitly specified as a mutual belief of all STAPLE agents so that the belief reasoner can use it.

4.3.1 PWAG Interpreter

The modal interpreter for PWAG is similar to the interpreter for PGOAL and INTEND, and it works by following the formal definition of PWAG (Definition 2.8) that we restate here for convenience.

$$\begin{aligned}(\text{PWAG x y } p\ q) \triangleq\ & [\neg(\text{BEL x } p)\ \wedge\ (\text{PGOAL x } p\ q)] \vee \\ & [(\text{BEL x } p)\ \wedge\ (\text{PGOAL x } (\text{MB x y } p)\ q)] \vee \\ & [(\text{BEL x } \Box\neg p)\ \wedge\ (\text{PGOAL x } (\text{MB x y } \Box\neg p)\ q)] \vee \\ & [(\text{BEL x } \neg q)\ \wedge\ (\text{PGOAL x } (\text{MB x y } \neg q))]\end{aligned}$$

This definition states that an agent x has a PWAG towards another agent y when the following holds: if agent x believes that p is not currently true then it will have a persistent goal to achieve p, and if it believes p to be either true, or to be impossible, or if it believes the relativizing condition q to be false, then it will have a persistent goal to bring about the corresponding mutual belief with agent y. Note that the PGOAL in the first three disjuncts are relative to q and therefore, establishing a mutual belief that $\neg q$ allows the PWAG to be dropped. From the above definition, we see that the PWAG interpreter needs to reason about mutual beliefs. As such, we enhance the belief reasoner by adding rules as in Table 4.4 to support mutual beliefs. Recall that the belief reasoner can only evaluate what the agent believes. Therefore, the reasoning rules for mutual beliefs in Table 4.4 are with respect to the agent's belief modality. As noted earlier, the current research does not explicitly focus on the complexity issues inherent in belief reasoning but rather chooses to build upon existing literature on this topic [22, 9, 59].

[15] A previous implementation of STAPLE did interpret both JPG and JI directly rather than interpreting PWAG.

We also modify the consistency checker to check if a given PWAG is inconsistent with any existing PWAG, PGOAL or INTEND of the agent. First, the currently active disjuncts in the definition of PWAG are found by invoking the belief reasoner in a Prolog engine over the agent's belief base to test whether $\neg$(BEL self p) or (BEL self p) or (BEL self $\Box\neg p$) or (BEL self $\neg q$) is true. Thereafter, the PGOAL conjunct of the currently active disjunct is taken to be the active PGOAL, and the consistency checker tests whether this active PGOAL is inconsistent with any existing PWAG, PGOAL, and INTEND of the agent. A PWAG is assumed to be inconsistent with another PWAG at any given time if the active PGOALs for each PWAG at that time are inconsistent with each other. If a new PWAG is found to be inconsistent with existing PGOAL, INTEND, or PWAG then the agent does not adopt the new PWAG, that is, it is not pushed onto the commitment stacks.

Table 4.4: Sample Deduction Rules for Mutual Belief

(BEL self p) :- (MB self y p)
(BEL self (BEL y p)) :- (MB self y p)
(BMB self y p) :- (MB self y p)
(BMB y self p) :- (MB self y p)
(BEL self (MB self y (BEL self p))) :- (MB self y p)
(BEL self (MB self y (BEL y p))) :- (MB self y p)
(BEL self (MB self y p∧q)) :- (BEL self (MB self y p)) ∧ (BEL self (MB self y q))
(BEL self (MB self y □¬(p∧q))) :- (BEL self (MB self y □¬p)) ∨ (BEL self (MB self y □¬q))

As in the case of PGOAL and INTEND, the consistency check is done before pushing a new PWAG on the stack. Thereafter, appropriate triggers are set depending on the context for conditions that may result in switching the currently active disjunct as well in discharging the PWAG itself. For instance, if the first disjunct is active, then triggers are set for (BEL self p), (BEL self $\Box\neg p$), and (BEL self $\neg q$) as well as for (MB self y p), (MB self y $\Box\neg p$), and (MB self y $\neg q$). When one of the first three triggers fires then re-evaluating the PWAG results in a different active PGOAL that the agent then pursues. The firing of any of the last three triggers results in discharge of the PWAG. Similarly, if the second disjunct in the definition of PWAG is active then triggers are set only for the last two disjuncts, that is, for (BEL self $\Box\neg p$), (BEL self $\neg q$), (MB self y $\Box\neg p$), and (MB self y $\neg q$). Triggers are not set here for the achievement and impossibility of the currently active PGOAL because that is taken care of during interpretation of that PGOAL. Note

that a trigger for (MB self y p) actually tests whether or not the term (BEL self (MB self y p)) can be deduced from the agent's belief base, that is, it is a test for (BMB self y p).

Once a PWAG is on the agent's stack, the currently active PGOAL for that PWAG is computed and subgoaled. When an active PGOAL of a PWAG is achieved, or is found to be impossible, or irrelevant, the PWAG is re-evaluated, thereby, either discharging the PWAG or leading to a different active disjunct, and consequently a different PGOAL to be achieved as per Definition 2.8. Note that the execution of PWAG results in mutual beliefs to be established as needed for the achievement, impossibility, and irrelevance of the committed goal. As such, when two agents have PWAG towards each other for achieving the same proposition, the resulting behavior is exactly the same as if the agents were jointly committed to each other for achieving that goal.

4.3.2 Implementing Communicative Acts in STAPLE

The communicative actions in STAPLE are just like any other action and so they must have at least a *code* and an *effect*. Additionally, STAPLE communicative actions may have a list of *desired effects*. These three components of a communicative act have the following functionality in STAPLE:

1. The code of a communicative action simply constructs a message and asks the communication manager (Figure 4.1) to send it to the intended recipient over the network. The communication manager encapsulates the low level details of transmitting the message appropriately.

2. From the semantics of communicative acts defined in terms of attempt (Section 2.5), recall that a communicative act has an associated intention and an associated goal. The intention associated with a communicative act is the minimum outcome that an agent intends to bring about via the performance of that communicative act. If the agent has reasons to believe that his intention associated with performing a communicative act was not achieved then he may retry that communicative act. As such, the effect a communicative action is set to be the *intention* part in the semantics of the communicative act. The (intended) effect of a communicative is used by the STAPLE interpreter for action selection during means-end reasoning.

3. The desired effect of a communicative action is set to be the *goal* part in the semantics of that communicative act. The desired effect is used by the STAPLE interpreter for

action selection during means-end reasoning only if it is unable to find any actions whose effect unifies with the goal to be achieved.

To summarize, the intention associated with a communicative act is used to specify the effect of the corresponding action in STAPLE, and its associated goal is used to specify the desired effect of that action. Also, we assume that the performance of communicative actions does not change the proposition being communicated (Assumption 2.2). Therefore, if an agent believed p right before performing a communicative act that p then it also believes p right after performing that communicative act. We also assume that agents are sincere in their communication (Proposition 2.5). Its consequence is that if an agent performs a communicative act intending that the recipient come to believe p as a result of the communicative act then that agent must itself believe p. As a result of these assumptions, we can get rid of the "BEFORE" and "AFTER" predicates in the intended and desired effects of the communicative acts in Section 2.5. Another simplification is that all facts in an agent's belief base are interpreted as if the agent believes those facts. Therefore, a unilateral mutual belief (BMB – meaning that an agent believes there is mutual belief) predicate in the effect and desired effect attributes can be re-written as mutual belief (MB). The precondition of the request communicative act is that the requestee does not already believe the propositions about which the requester intends to establish mutual belief via the request.

Basic communicative acts

Theorem 2.3 is used to define the INFORM action in STAPLE as an action whose effect is (MB x y (BEL x p)). The executable code for that action first requests the communication manager to connect to the agent being informed (if not already connected), after which it constructs a message and asks the communication manager to send that message to the intended recipient. The communicative action CONFIRM is defined similar to that of an INFORM with a different precondition. Similarly, Theorem 2.2 is used to define the REQUEST action in STAPLE to be an action whose effect is (MB x y (PWAG x y ϕ q)) where ϕ is the goal of the REQUEST in Definition 2.13. The definitions of these communicative acts are listed in Table 4.5.

Table 4.5: Definitions of Basic Communicative Acts in STAPLE

```
action_definition(request,4) :-
  [args:  [X,Y,A,Q], %Requester, Requestee, Action, Relativizing Condition
  code:  {% code to compose and send message},
  precondition:  {(\+ Y=self) ∧ ∼bel(X,done(A))
        ∧ ∼bel(X,pwag(Y,X,done(A),pwag(X,Y,done(A),Q)∧Q))
        ∧ ∼bel(X,bel(Y,pwag(X,Y,done(A)∧pwag(Y,X,done(A),
                                          pwag(X,Y,done(A),Q)∧Q),Q)))},
  desired_effects:  [<>done(A),
          <>pwag(Y,X,done(A),pwag(X,Y,done(A),Q)∧Q)],
  effects:  [(mb(X,Y,pwag(X,Y,done(A)∧pwag(Y,X,done(A),
                                       pwag(X,Y,done(A),Q)∧Q),Q)),1.0)]
  ].

action_definition(inform,3) :-
  [args:  [X,Y,P], %Informer, Informee, Informed Proposition
  precondition:  {bel(X,P) ∧ ∼bel(X,bel(Y,P)) ∧ (\+ P=bel(X,_P))}
  code:  {% code to compose and send message},
  desired_effects:[<>mb(X,Y,P)]
  effects:  [(mb(X,Y,bel(X,P)),1.0)]
  ].

action_definition(confirm,3) :-
  [args:  [X,Y,P], %Informer, Informee, Informed Proposition
  precondition:  {bel(X,P) ∧ (∼bel(X,bel(Y,bel(X,P))))},
  code:  {% code to compose and send message},
  desired_effects:[<>mb(X,Y,P)]
  effects:  [(mb(X,Y,bel(X,P)),1.0)]
].
```

Plans to establish mutual belief

The definitions of the communicative acts listed above are used to specify plans to establish mutual belief in STAPLE agents. For instance, the body of a STAPLE plan for establishing mutual belief that p is an action expression consisting of an action to achieve (MB self y (BEL self p)) followed by an OR expression that terminates if it is mutually believed that this agent is competent with respect to p, else it waits for the other agent to establish (MB self y (BEL y p)). Note that these actions and the plan are the same as other STAPLE actions and plans, and the STAPLE interpreter reasons about them just like any other actions and plans. However, now there can be actions whose successful performance may lead to the agent acquiring a PWAG or a PGOAL. For instance, a STAPLE agent must have the (PWAG x y ϕ q), where ϕ is from Definition 2.13 of REQUEST, just after requesting another agent y to do an action a. This is implemented by adding a declarative rule to the STAPLE rule base that is applicable only when an action is successfully performed (recall that the INTEND interpreter executes applicable rules whenever an intended action either succeeds or fails). This rule checks if the effect of the action that was just performed leads to a commitment (either PGOAL or PWAG), and if so, the agent adopts that commitment if it does not already have it.

Composed communicative acts

Any number of communicative actions may be defined as needed by composition from the primitive communicative acts using action formation operators and using specialized content. One particularly useful communicative act available to all STAPLE agents is the AGREE communicative act (AGREE x y e a q t) where an agreeing agent x informs the listening agent y that he has a PWAG with respect to y to perform action a with respect to both y's PWAG that x do a relative to q, and q. It is shown in Theorem 2.4, that successful performance of an AGREE communicative act establishes mutual belief by default that the sender x has the specified PWAG towards the recipient y. Therefore, the AGREE action in STAPLE is defined as an action having the default effect (MB x y (PWAG x y (DONE x a) (PWAG y x (DONE x a) q)$\wedge q$)). As such, when an agent receives an AGREE in response to its REQUEST, the belief base maintenance system for this agent asserts the JPG resulting from this exchange of messages using Theorem 2.1 on interlocking PWAGs. Similarly, the recipient of a REQUEST asserts the same JPG in its belief base after performing an AGREE. In fact, the action expression consisting of REQUEST followed by waiting for the effect of AGREE is the body of a STAPLE plan

to establish JPG.

Table 4.6: Definitions of Composed Communicative Acts in STAPLE

```
action_definition(agree,3) :-
    [args:  [X,Y,P], %Informer, Informee, Informed Proposition
    precondition:  {bel(X,P) ∧ ∼bel(X,bel(Y,P)) ∧ (\+ P=bel(X,_P))}
    code:  {% code to compose and send message},
    effects:  [(mb(X,Y, pwag(Y,X,done(A),pwag(X,Y,done(A),Q)∧Q)),1.0)]
    ].
action_definition(refuse,3) :-
    [args:  [X,Y,P], %Informer, Informee, Informed Proposition
    precondition:  {bel(X,P) ∧ ∼bel(X,bel(Y,P)) ∧ (\+ P=bel(X,_P))}
    code:  {% code to compose and send message},
    effects:  [(mb(X,Y, '[]'(∼pwag(X,Y,done(A),pwag(Y,X,done(A),Q)∧Q))),1.0)]
    ].
```

The REFUSE communicative act is defined similar to that of AGREE using Definition 2.16. The definitions of AGREE and REFUSE listed in Table 4.6 do not have the desired effects because the effect and desired effect of these communicative acts are logically equivalent.

Other composed communicative acts

STAPLE also defines several other composed communicative actions that were used in the examples in Chapter 3. These communicative actions are listed in Table 4.7.

INFORM-IF is defined as an INFORM of p or INFORM of $\sim p$ depending on whether the informer believes p or $\sim p$. INFORM-REF is similar to inform except that what is being informed is the referent c that makes p true. In this definition, the quantifier i is the Russelian quantifier 'iota'. ASK is defined as a REQUEST to do an INFORM-REF and ASK-IF is defined as a REQUEST to do an INFORM-IF. Note that the above definitions only illustrate various ways in which communicative acts can be composed in STAPLE. They encode one possible definition of these composed communicative acts from the multi-agent systems literature.

So far, we have focused only on singleton communicative actions. However, we need to examine the issues involved in the joint execution of complex action expressions by the agents involved before exploring action expressions consisting of multiple communicative acts.

Table 4.7: Other Composed Communicative Acts in STAPLE

```
action_definition(informif,3) :-
      [args:  [X,Y,P],
      precondition:  {nonvar(P) ∧ (bel(X,P) ∨ bel(X,∼P))},
      code:  {% code to compose and send appropriate inform message},
      effects:  [(mb(X,Y,bel(X,Q)),1.0)] %Q is either P or ∼P
      ].

mb(X,Y,done(informif(I,J,P))) :- mb(X,Y,done(inform(I,J,P)));
                                 mb(X,Y,done(inform(I,J,∼P))).

action_definition(informref,4) :-
      [args:  [X,Y,C,i(Z,P)],
      precondition:  bel(X,equals(C,i(Z,P))),
      code:  {% code to compose and send message},
      effects:  [(mb(X,Y,bel(X,equals(C,i(Z,P)))),1.0)]
      ].

plan(askref,3) :-
      [args:  [X,Y,i(Z,P)], %Sender, Recipient, i(Z,P(Z))
      precondition:  bel(self,exists(K,bel(Y,equals(K,i(Z,P))))),
      body:  {request(X,Y,action(informref(Y,X,C,i(Z,P)),Y),Q)},
      effects:  [(exists(J,bel(self,equals(J,i(W,P)))),1.0)]
      ].

plan(askif,3) :-
      [args:  [X,Y,P], %Sender, Recipient, Proposition to inquire
      precondition:  bel(X, bel(Y,P) ∨ bel(Y,∼P)),
      body:  {request(X,Y, action(informif(Y,X,P),Y), Q)},
      effects:  [(bel(X,P) ∨ bel(X,∼P),1.0)]
      ].
```

4.3.3 Executing Action Expressions Jointly

A joint action expression is an action expression that involves multiple cooperating agents and therefore, each action in that action expression may have a different actor. A singleton action can have only one actor in our framework. Hence, doing an action jointly, say, lifting a table by two agents at the same time is considered to be composed of two different actions, one by each agent, being done concurrently. Executing joint action expressions by two agents reduces to each agent's interpreting their individual PGOAL for that action expression. As such, the theorems in Table 2.1 are still applicable but there are several new issues that need to be accounted for in the joint case.

We modify the action representation to specify the agent of the action because action expressions can now involve actions to be done by different agents in a team. An action a should now be specified as action(actor, a) otherwise a default actor such as "any" or "team" is added depending on the context when an action expression is parsed. The event identifier for an action is also modified to take into account the actor of the action. The default STAPLE rule is to intend a plan for task allocation among team members when 'team' is specified as the actor for an action, provided that such a plan is available. Otherwise, the actor 'team' is replaced by 'all' meaning that all team members have to perform that action. An agent programmer can make use of the research on distributed task allocation algorithms in multi-agent systems to provide appropriate plans.

The joint intention theory prescribes that mutual belief be established in the team for achievement, impossibility or irrelevance of the entire jointly committed action expression. Teams are free to devise their own policies to establish mutual belief while the joint action expression is being executed. The default policy in STAPLE to execute joint actions in lockstep [70] is implemented as a declarative rule to establish mutual belief in a team about an action that the agent just did as part of the joint action.

As mentioned earlier, an agent x may end up with a term like (PGOAL x (DONE action(y, a)) q) on top of one of its stacks meaning thereby that agent x is committed to agent y's doing the action a. Rules are used to decided what to do in such a situation. The default rule searches for a plan for being helpful and cooperative towards agent y in the present situation and instantiates it if such a plan is found. The plan to be helpful and cooperative is domain dependent[16] and must be provided by the agent programmer. For example, if action(y, a) is the action of carrying a lamp outside the room by agent

[16]One domain independent way to be helpful is to look for a precondition that is false for the other agent and adopt it as its own goal. However, this plan is not yet implemented in the current interpreter.

y then a plan for agent x to be helpful and cooperative may include agent x to open the door proactively if the door is closed. If a plan to be helpful and cooperative is not found, then agent x intends to wait for agent y to attempt to establish mutual belief that (DONE action(y, a)). In order to prevent agent x from waiting forever, before adopting the intention to wait for the mutual belief to be established, agent x must believe that agent y has an individual commitment to do a, and will have a commitment to establish mutual belief after doing a. This condition is trivially satisfied if it is mutually believed that agent y has a PWAG towards agent x either for (DONE action(y, a)) or for a complex action expression that consists of action(y, a). Otherwise, a joint commitment will need to be established, say, by using a request protocol. Alternatively, agent x may be provided with a plan that decides how long that agent should wait for agent y to finish doing the action a.

Non-deterministic OR is inherently problematic when executed jointly because random selection of one of the actions as in the case of single agents will not work as different agents may end up selecting different actions. Possible solutions include negotiation among the team members to decide which action to choose. The default STAPLE implementation uses random selection of one of the actions by a *team leader* who then establishes mutual belief about the chosen action. If the actor of the selected action establishes mutual belief about impossibility of performing that action, then the leader chooses another action from the remaining untried actions. This algorithm continues until either one of the actions succeeds or all the actions in the OR expression are mutually believed to be impossible. In the current STAPLE implementation, the agents mutually believe that the team leader is the agent who initiated the joint commitment, for instance, the agent who performed the initial request that resulted in establishing the joint commitment. This leadership-based default algorithm is used only when the actors for the actions in the OR expression are different agents – if the same agent is the actor for all the actions in the OR expression that is being executed jointly, then that agent decides which action to choose as if it were executing that OR expression just by itself.

As an aside, during execution of joint actions, there may be actions done that were not specified in the original action expression (such as establishing mutual belief caused by the lockstep policy, or executing a plan to achieve the preconditions of another plan that performs a required action). Further, it is difficult to exactly time-align actions by two different agents in a sequence. As such, the action sequence operator in STAPLE is at best an approximation of the usual sequence operator in dynamic logic, and it may be

regarded as specifying only a partial ordering of actions. One way to address this difficulty is to define a weak-sequence operator[17], add its semantics to the model in [25], and use it in place of the sequence operator to be faithful to the actual semantics of the operators.

The present dissertation treats multi-agent conversations as joint actions involving multiple communicative acts between multiple agents. From this perspective, a conversation protocol is just like any other joint action expression and therefore, it can be executed like other action expressions. We present a formalism for conversation protocols as joint action expressions and discuss its implementation in STAPLE in Chapter 8.

4.4 SUMMARY

We have presented the implementation of an interpreter for directly executing agent specifications in the logical language of joint intentions. The STAPLE interpreter attempts to satisfy a formula such as PGOAL using its definition in the JI theory. Thus, this direct execution of specifications can be viewed as implicit construction of a model for the logical formulae under reasonable assumptions. The assumptions are necessary because logic as expressive as that used in STAPLE represents an ideal abstraction but the real implementations invariably imposes various constraints. The question then is how do we ensure that the interpreter is faithful towards the logic that it attempts to interpret? One feasible strategy is to treat the model theory of the logic as specifying the denotational semantics of the logical language, specify an operational semantics of the language based on the denotational semantics, and use the operational semantics for the actual implementation. This strategy is the one that we have chosen to follow in this dissertation. The operational semantics implemented by the STAPLE interpreter is presented in the appendix.

[17] A model theoretic definition of the weak sequence $a; b$ would be one in which the action b does not necessarily occur immediately after action a. The intervening actions allowed to occur between a and b may be further constrained to be those that lead to establishment of mutual beliefs.

Chapter 5

Extending the Theory of Teamwork and Communication to Support Groups

Artificial as well as human agents not only interact with individual agents, but they also need to coordinate and communicate with groups of agents. We have so far introduced and analyzed teams consisting of just two agents. However, many teams in the real world consist of multiple coordinating agents. Some of these teams even exist independent of the identity of their members. For example, the 'Yankees' remain the same team even if all its members are traded. Moreover, many of these teams do not get dissolved once their current goal is achieved but are reused again and again to achieve other team goals. For example, the same Marine team can be used for multiple reconnaissance missions. Our supporting these resilient teams requires our extending the joint intention theory in Chapter 2 to use groups and named teams instead of its constituent agents. Furthermore, the communicative acts themselves need to be redefined for creating and discharging teams of multiple agents. This chapter introduces the notion of dynamic but persistent teams, provides a means to specify the formal semantics of group communication, and finally it describes the communicative acts and accompanying results that are used in an implemented version of STAPLE. These extensions to the teamwork theory are one of the contributions of this dissertation.

5.1 PERSISTENT AND DYNAMIC TEAMS

We use the definitions of persistent and dynamic teams that follow to specify and implement the fault-tolerance behavior of AAA brokers in the next chapter. Thereafter, we declaratively specify the same fault-tolerance behavior (using these definitions) for brokers written in STAPLE and observe that it results in the STAPLE brokers' having the same

fault-tolerance behavior as that of the AAA brokers (Chapter 7).

We have seen in Chapter 2 that team activity is explained in terms of the theory of joint intentions. This theory characterizes an agent's behavior in a team in terms of its internal state described in modal logic, linear time temporal logic, and dynamic logic of action. Recall that a joint persistent goal (*JPG*) expressed in terms of weak mutual goal (*WMG*) formalizes the notion of joint commitment. The existence of a JPG between two agents is a sufficient condition for the formation of a team with respect to that JPG. From Definition 2.6 of JPG, we have

$$\begin{aligned}(\text{JPG x y } p\ q) \triangleq\ & (\text{MB x y } \neg p) \wedge (\text{MG x y } p) \wedge \\ & (\text{UNTIL } [(\text{MB x y } p) \vee (\text{MB x y } \Box\neg p)\ \vee (\text{MB x y } \neg q)] \\ & \quad (\text{WMG x y } p\ q))\end{aligned}$$

It is apparent from the above definition of JPG that the JPG becomes invalid when either x or y is no longer available. Even the individual commitments implied by this JPG is allowed to be dropped because it is now impossible to establish any of the mutual beliefs in the *until* clause. Moreover, the JPG is no longer valid when the agents mutually believe p and hence the team cannot exist after the mutual goal is achieved and is mutually believed, even if there are residual commitments. However, teams in the real world may be one-time teams that are disbanded after the team goal has been achieved, or they may be persistent teams that continue to exist even when the team members change. Persistent teams are especially desirable from a fault-tolerance perspective because agents that fail will generally be replaced by other agents during the recovery process. Next, we introduce a notation for representing groups, redefine the basic concepts along with JPG for groups, and introduce a notion of team commitment based on maintenance goals.

5.1.1 Representing Groups

We consider a group to be a collection of entities, that are defined by a membership property. This can be captured by a predicate consisting of a free variable that ranges over individuals and, in general, ranges over subgroups as well. We use a special notation to represent propositions that refer to a group whose members satisfy certain properties. Let α_z be a formula with free variable z and let π(z) be a one-place predicate with variable z. We call $\alpha_z(\pi(\text{z}))$ a quasi-formula with respect to π(z) and use the notation φ to denote a formula defined by the following rule:

1. If φ is a formula, then $\langle\varphi\rangle = \varphi$

2. If φ is the quasi-formula $\alpha_z(\pi(\mathrm{z}))$, and $\alpha_z(\pi)$ is the formula that results by substituting z for $\pi(\mathrm{z})$ in φ, then $\langle\varphi\rangle = \forall \mathrm{z}.\ \pi(\mathrm{z}) \supset\ \alpha_z(\pi)$

For example,

$\langle$BEL x $p\rangle$ = (BEL x p)
$\langle$BEL λz.(member z τ) $p\rangle = \forall$z (member z τ) $\supset$ (BEL z p)
$\langle$BEL λy.(member y τ) $\langle$BEL λx.(member x τ) $p\rangle\rangle$
$= \forall$y (member y τ) $\supset$ (BEL y [$\forall$x (member x τ) $\supset$ (BEL x p)])

The last example says that everybody who is a member of τ believes that *whoever* is a member of τ believes p. In these examples, note that this belief does not involve quantifying-in. For the purpose of this dissertation, we can assume the existence of a "member" predicate such that the proposition (member z τ) is true *iff* z is a member of the group τ. Halpern and Moses [48] have an E^k operator for similar semantics but since we are concerned with the group membership predicate and are treating groups as individuals, we have chosen to extend the language explicitly.

5.1.2 Mutual belief and mutual goal in a group

It is possible to define mutual belief in several different ways corresponding to different possible definitions of unilateral mutual belief or BMB. We explore various definitions of group belief and unilateral group belief later in this chapter. Unless explicitly mentioned otherwise, we will assume *extensive* unilateral belief in this dissertation – an agent believes p and believes that everybody believes p and believes that everybody believes that everybody believes p and so on.

Definition 5.1. Group BMB
An agent z has (extensive) unilateral mutual belief with a group τ about a proposition p when it has a BMB with every member of the group τ about the proposition that every member of the group believes p.

(BMB z τ p) $\triangleq$ $\langle$BMB z λy.(member y τ) $\langle$BEL λx.(member x τ) $p\rangle\rangle$

Definition 5.2. Group Mutual Belief
A group τ mutually believes p when all the members of the group have (extensive) unilateral mutual belief with the group about p.

(MB τ p) $\triangleq$ $\langle$BMB λz.(member z τ) τ $p\rangle$

Definition 5.3. Group Mutual Goal
A group τ has p as a group mutual goal if the group mutually believes that everybody in the group has p as an individual goal

$$(\text{MG } \tau\ p) \triangleq (\text{MB } \tau\ \langle \text{GOAL } \lambda \text{z.(member z } \tau)\ \Diamond p \rangle)$$

We now use these concepts to define team commitments that result in persistent teams.

5.1.3 Joint Commitment Revisited

We overload the definitions of joint persistent goal (JPG) and its related concepts to use a team as an entity instead of the individual agents in the team. Additionally, we will assume that the team-members have beliefs about team membership at all times in order to have correct group mutual beliefs.

Definition 5.4. Team Weak Achievement Goal
A member z of a team τ is said to have a weak achievement goal (WAG) with respect to the team when the following conditions hold:

1. If the team member believes that the goal p is not yet achieved, it has an individual goal to eventually bring about p.

2. If it believes that the goal has been achieved, is impossible to achieve, or is irrelevant then it has an individual goal to bring about the corresponding group mutual belief.

$$\begin{aligned}(\text{WAG z } \tau\ p\ q) \triangleq\ & [\neg(\text{BEL z } p) \wedge (\text{GOAL z } \Diamond p)] \vee \\ & [(\text{BEL z } p) \wedge (\text{GOAL z } \Diamond(\text{MB } \tau\ p))] \vee \\ & [(\text{BEL z } \Box\neg p) \wedge (\text{GOAL z } \Diamond(\text{MB } \tau\ \Box\neg p))] \vee \\ & [(\text{BEL z } \neg q) \wedge (\text{GOAL z } \Diamond(\text{MB } \tau\ \neg q))]\end{aligned}$$

Definition 5.5. Team Weak Mutual Goal
A team τ has p as a weak team mutual goal if the team mutually believes that everybody in the team has p as a weak team goal.

$$(\text{WMG } \tau\ p\ q) \triangleq (\text{MB } \tau\ \langle \text{WAG } \lambda \text{z.(member z } \tau)\ \tau\ p\ q \rangle)$$

Definition 5.6. Team Joint Persistent Goal
A team τ has a team persistent goal or team commitment when the following conditions hold:

1. The team mutually believes that the goal p has not been achieved.
2. The team has a mutual goal to achieve the goal p.
3. The team has p as a weak team mutual goal until there is mutual belief about the completion, feasibility or irrelevance of p.

$$(\text{JPG } \tau\ p\ q) \triangleq (\text{MB } \tau\ \neg p) \wedge (\text{MG } \tau\ p) \wedge \\ (\text{UNTIL } [(\text{MB } \tau\ p) \vee (\text{MB } \tau\ \Box\neg p) \vee (\text{MB } \tau\ \neg q)]\ (\text{WMG } \tau\ p\ q))$$

Given that the above JPG is defined in terms of "whoever is a member of the team", there will be instances when a team member does not know about the other team members. However, this is not a problem as long as the team members are able to communicate with "whoever" is in the team at a given instance irrespective of whether or not they know all those members. In fact, this property is a characteristic of many real life examples of teamwork such as remote collaboration on a project where not everybody at each location knows everybody else at the other location. However, to simplify group communication between team members and to make sure that there is appropriate mutual belief in the team when agents are allowed to join and leave a team, we will assume that every team member eventually knows every other team member.

Assumption 5.1. *Dynamic Team Assumption*

If a team member privately comes to believe that the group has a new member, or an existing member is no longer in the team then it has an individual goal to bring about the corresponding group mutual belief. Formally, we define the property (dynamic τ) as follows.

$$(\text{dynamic } \tau) \triangleq \\ \forall \text{z } (\text{member z } \tau) \supset \{\ [\forall \text{y } \text{Q}_{z,y} \supset (\text{GOAL z } \Diamond(\text{MB } \tau \neg(\text{member y } \tau))] \wedge \\ [\forall \text{y } \text{P}_{z,y} \supset (\text{GOAL z } \Diamond(\text{MB } \tau\ (\text{member y } \tau))]\ \}$$

where,

$$\text{Q}_{z,y} = (\text{BEL z } \exists e\{\text{DONE } (\text{member y } \tau)?;e;\neg(\text{member y } \tau)?\})$$
$$\text{P}_{z,y} = (\text{BEL z } \exists e\{\text{DONE } \neg(\text{member y } \tau)?;e;(\text{member y } \tau)?\})$$

and e is a singleton event.

Unless stated otherwise, we will assume the above property about team membership in the remainder of this dissertation

Definition 5.7. Team Joint Intention

A team jointly intends an action a if there is a team commitment to do the action while believing that the team is going to do the action next

$$(\text{JI } \tau \ a \ q) \triangleq [\text{TPG } \tau \ (\text{DONE } \tau \ (\text{MB } \tau \ (\text{HAPPENS } \tau \ a))?;a) \ q]$$

Joint commitment and joint intention are defined above with respect to the team as an entity. This allows a team having these commitments to continue as long as there is no mutual belief about the completion, impossibility or irrelevance of the team goal even if some of the team members leave and new members join the team. The achievement of mutual belief among team members requires group communication that we explore later in this chapter. Next, we use maintenance goals to characterize another important property of a persistent team – the ability to continue beyond one-time achievement goals.

5.1.4 Maintenance Goal

Maintenance goals can be restorative or preventive. We will only be concerned with restorative maintenance goals in the fault-tolerance examples in the next chapter. However, we note that preventive maintenance goals can also play an important role in achieving fault tolerance in multi-agent systems by attempting to prevent events that can potentially cause failure.

Definition 5.8. Restorative Maintenance Goal

An agent has a (restorative) maintenance goal if the following is true of the agent: if the agent does not believe p, it will adopt the goal that p be eventually true. The maintenance goal is persistent (PMtG) if this fact remains true of the agent at least until the agent either believes that it is impossible to maintain p or that the maintenance goal is irrelevant.

$$\begin{aligned}(\text{PMtG x } p \ q) \quad \triangleq \quad & [\neg(\text{BEL x } p) \supset (\text{GOAL x } \Diamond p)] \wedge \\ & (\text{UNTIL } [(\text{BEL x } \Box\neg p) \vee (\text{BEL x } \neg q)] \ [\neg(\text{BEL x } p) \supset (\text{GOAL x } \Diamond p)])\end{aligned}$$

Theorem 5.1. *If an agent who has a persistent maintenance goal for p comes to believe $\neg p$ then it will adopt a persistent achievement goal (PGOAL) for p in addition to the persistent maintenance goal. Formally,*

$$\models (\text{PMtG x } p \ q) \wedge (\text{BEL x } \neg p) \supset (\text{PGOAL x } p \ q)$$

Proof. We need to show that all three conjuncts in the definition of PGOAL (Definition 2.1) follow from the premise of the theorem.

(1) (BEL x $\neg p$) [From premise]

(2) To establish that (GOAL x $\Diamond p$):

(a) $\neg$(BEL x p) [(BEL x $\neg p$) $\supset$ $\neg$(BEL x p)]

(b)[$\neg$(BEL x p) $\supset$ (GOAL x $\Diamond p$)] [From premise and Definition 5.8]

(c) (GOAL x $\Diamond p$) [From (a) and (b) using Modus Ponens]

(3) To establish the UNTIL conjunct in the definition of PGOAL, we use the reasoning that the consequent of an implication must remain true at least until either the antecedent of the implication or the implication itself is no longer true.

(d) The implication [$\neg$(BEL x p) $\supset$ (GOAL x $\Diamond p$)] must remain true at least until [(BEL x $\Box\neg p$)$\vee$(BEL x $\neg q$)] is true. [From Definition 5.8]

Therefore, the consequent (GOAL x $\Diamond p$) of this implication must remain true at least until

[(BEL x $\Box\neg p$)$\vee$(BEL x $\neg q$)] is true. [The antecedent $\neg$(BEL x p) is true]

(e) The consequent (GOAL x $\Diamond p$) of the implication [$\neg$(BEL x p) $\supset$ (GOAL x $\Diamond p$)] must remain true at least until the antecedent $\neg$(BEL x p) is no longer true.

[Assuming that the implication itself remains true]

Therefore, (GOAL x $\Diamond p$) must remain at least until [(BEL x p) $\vee$ (BEL x $\Box\neg p$)$\vee$(BEL x $\neg q$)] is true. [From (d) and (e)]

This establishes the desired result.

□

Definition 5.9. Weak Team Maintenance Goal

Weak team maintenance goal (*WTMtG*) is defined analogous to weak team goal. If an agent comes to believe p or $\neg p$ or impossibility or irrelevance of p then it will adopt a goal to bring about the corresponding mutual belief if that mutual belief does not already exist.

$$
\begin{aligned}
\text{(WTMtG x } \tau\ p\ q) \triangleq\ & [\neg(\text{BEL x } p) \supset\ (\text{GOAL x } \Diamond p)] \wedge \\
& [(\text{BEL x } \neg p) \wedge \neg(\text{MB } \tau\ \neg p) \supset (\text{GOAL x } \Diamond(\text{MB } \tau\ \neg p))] \wedge \\
& [(\text{BEL x } p) \wedge \neg(\text{MB } \tau\ p) \supset (\text{GOAL x } \Diamond(\text{MB } \tau\ p))] \wedge \\
& [(\text{BEL x } \neg q) \wedge \neg(\text{MB } \tau\ \neg q) \supset (\text{GOAL x } \Diamond(\text{MB } \tau\ \neg q))] \wedge \\
& [(\text{BEL x } \Box\neg p) \wedge \neg(\text{MB } \tau\ \Box\neg p) \supset (\text{GOAL x } \Diamond(\text{MB } \tau\ \Box\neg p))]
\end{aligned}
$$

Definition 5.10. Weak Team Mutual Maintenance Goal
A team has a weak team mutual maintenance goal (WTMMtG) of p *iff* the team mutually believes that everybody who is a member of the team has a weak team maintenance goal of p towards the team.

$$(\text{WTMMtG } \tau\ p\ q) \triangleq (\text{MB } \tau\ \langle \text{WTMtG } \lambda z.(\text{member z } \tau)\ \tau\ p\ q\rangle)$$

Definition 5.11. Team Maintenance Goal
A team of agents has a team maintenance goal (TMG) for p iff it has a WTMMtG for p and this WTMMtG persists at least until the team mutually believes the impossibility or the irrelevance of maintaining p.

$$\begin{aligned}(\text{TMtG } \tau\ p\ q) \quad \triangleq \quad & (\text{WTMMtG } \tau\ p\ q) \wedge \\ & (\text{UNTIL } [(\text{MB } \tau \Box\neg p) \vee (\text{MB } \tau \neg q)]\ (\text{WTMMtG } \tau\ p\ q))\end{aligned}$$

The above definition of team maintenance goal leads to a team commitment to achieve a proposition whenever the team mutually believes that the proposition committed to is not true. The next theorem establishes this result.

Theorem 5.2. *If a team having TMtG for p comes to mutually believe $\neg p$ then it will adopt a JPG for p. Formally,*

$$\models [(\text{TMtG } \tau\ p\ q) \wedge (\text{MB } \tau \neg p)] \supset (\text{JPG } \tau\ p\ q)$$

Proof. We want to show that all the conditions for JPG follow from the premise of the theorem.
(1) (MB τ $\neg p$) is true (assume that the antecedent is true).
(2) (MB τ $\neg p$) $\supset \forall$z (member z τ) $\supset$ (BEL z $\neg p$)
$\supset \forall$z (member z τ) $\supset \neg$(BEL z p)
Using the definition of WTMMtG,
(WTMMtG $\tau\ p\ q) \wedge$ (MB τ $\neg p$)
$\supset$ (MB τ $\forall$z (member z τ) $\supset$ (WTMtG z $\tau\ p\ q) \wedge \neg$(BEL z $\neg p$))
$\supset$ (MB τ $\forall$z (member z τ) $\supset$ (GOAL z $\Diamond p$))
= (MG τ $\Diamond p$)
(3) From the definition of TMtG, we see that WTMMtG must hold at least until [(MB τ $\Box\neg p) \vee$ (MB τ $\neg q$)] becomes true.
Therefore, the implications in the definition of WTMtG must hold at least until

[(MB τ $\Box\neg p$)$\vee$(MB τ $\neg q$)] becomes true.

Hence, we can conclude the following where z is an arbitrary member of the group τ.

(a) If $\neg$(BEL z p) is true and remains true then [$\neg$(BEL z p) $\wedge$ (GOAL z $\Diamond p$)] must remain true at least until [(MB τ $\Box\neg p$) $\vee$(MB τ $\neg q$)] becomes true.

(b) If (BEL z p) is true and remains true then [(BEL z p) $\wedge$ (GOAL z $\Diamond$(MB τ p))] must remain true at least until (MB τ p) becomes true or [(MB τ $\Box\neg p$)$\vee$(MB τ $\neg q$)] becomes true, that is, until [(MB τ p)$\vee$(MB τ $\Box\neg p$)$\vee$ (MB τ $\neg q$)] becomes true.

(c) If (BEL z $\Box\neg p$) is true and remains true then [(BEL z $\Box\neg p$)$\wedge$ (GOAL z $\Diamond$(MB τ $\Box\neg p$))] must remain true at least until [(MB τ $\Box\neg p$)$\vee$(MB τ $\neg q$) becomes true.

(d) If (BEL z $\neg q$) is true and remains true then [(BEL z $\neg q$)$\wedge$(GOAL z $\Diamond$(MB τ $\neg q$))] must remain true at least until [(MB τ $\Box\neg p$)$\vee$(MB τ $\neg q$)] becomes true.

From (a), (b), (c), and (d), and by the fact that at least one of the antecedents in the corresponding implications must be true at any time, we can conclude that the disjunction

[$\neg$(BEL z p)$\wedge$ (GOAL z $\Diamond p$)] $\vee$
[(BEL z p)$\wedge$ (GOAL z $\Diamond$(MB τ p))] $\vee$
[(BEL z $\Box\neg p$)$\wedge$ (GOAL z $\Diamond$(MB τ $\Box\neg p$))] $\vee$
[(BEL z $\neg q$)$\wedge$ (GOAL z $\Diamond$(MB τ $\neg q$))]

must be true at least until [(MB τ p)$\vee$ (MB τ $\Box\neg p$)$\vee$ (MB τ $\neg q$)] becomes true.

The above disjunction is the definition of (WAG z τ p q) where z is an arbitrary member of the group τ.

Therefore, we can conclude that (WMG τ p q) must be true at least until
[(MB τ p)$\vee$(MB τ $\Box\neg p$)$\vee$(MBτ $\neg q$)] becomes true.

From (1), (2), and (3), (JPG τ p q) is true. This proves the desired result. □

We use the above definitions and results to specify a fault-tolerance behavior of brokers in the Adaptive Agent Architecture discussed in the next chapter. The modified definition of JPG and the Dynamic Team Assumption are used in the STAPLE implementation in this dissertation. Next, we explore the semantics of group communication.

5.2 GROUP COMMUNICATION

Artificial as well as human agents need to communicate with groups of agents. For instance, we post messages to mailing lists and notice boards; participate in teleconferences and videoconferences; publish web pages and books; speak in meetings and classrooms; talk on radio and television; and advertise on pamphlets and banners. Agents will be

assuming some of these responsibilities from humans and will therefore, need to be able to reason and communicate about group concepts. Moreover, in open multi-agent systems, where agents come and go dynamically, it will become ever more likely that agents will not know exactly to whom they are sending information or from whom they are requesting aid. These are compelling reasons to investigate the development of support for group communication in multi-agent systems. It is no surprise, therefore, that a large number of distributed software systems inevitably use some incarnation of broadcasting and multicasting.

However, we observe that the major agent communication languages have either no provision or no well-defined semantics for group communication. For instance, in the FIPA ACL (Agent Communication Language), the only way to inform a set of agents is to inform them individually, one at a time. Furthermore, semantics of the FIPA communicative acts imposes the precondition that the sender has certain beliefs about the mental state of the (known) addressee. Consequently, there is no way to send messages to unknown agents – a typical scenario in broadcast communication.

KQML does offer several primitives, such as broadcast and recruit-all, that have group flavor but these primitives are merely shorthand for a request to do a series of other communicative acts. Proper semantics cannot be given to group requests such as "One of you, please, get me a slice of that pie." We may safely conclude that principled support for group communication in the widely used agent communication languages does not exist.

Group communication is not just about sending a message to a large number of agents at the same time. As mentioned earlier, sometimes the sender does not know the specific recipients of a message. A person who posts the notice "Beware of dogs" may not know who will read that message. So the semantics of a communication language should allow for intentions with respect to "whoever gets this message," while allowing for constraints on the intended recipients and identification of this constraint for correct illocutionary effect. Furthermore, the intended actor for a communication may be a subset of the recipients or a completely different set. By sending an email to the CSE101 mailing list requesting Becker to take the attendance in the next class, the instructor not only made a request to Becker to take attendance but also let the whole class know that she requested Becker to do it. Senders need not only be individuals but can also be groups. An invitation card from John and Betty is actually a request to attend from "them". Individuals may be viewed as singleton groups. Therefore, the same communication primitives should work both for individual and group communication. We believe that any general-purpose agent

communication language should be able to deal with these aspects of communication.

To summarize, we have argued that (1) Agent communication languages should support group communication where communication between individuals is a natural special case; (2) An agent communication language that supports group communication should account for the recipients being unknown, the sender being a group, and the intended actors being different from the recipients.

5.2.1 Constraints on Communication Languages

We believe that the properties of communication in human society should be an essential guiding principle in the design of agent communication languages. These properties are requirements that an agent communication language should address.

- *Addressee Constraint*: An ACL should support communication addressed to individuals as well as to groups. Moreover, a group may have a stable, known membership, as in a mailing list, or its membership may be unknown, as in a radio broadcast addressed to all listeners.

- *Sender Constraint*: An ACL should support communication sent by individuals as well as by groups. Typically, an individual acts on behalf of a group when the sender happens to be a group: for example, the invitation card from a couple, and an official letter from a company.

- *Recipient Constraint*: An ACL should support unintended recipients or over-hearers that are an inevitable part of group communication. For example, anybody may happen to read a notice addressed to CSE101 students on the school notice board. Similarly, an announcement on an airport public announcement system requesting Alfred Hopkins to meet someone at the bookstall may include everybody else who hears the announcement as an over-hearer.

- *Actor Constraint*: An ACL should support intended actors being wholly different from either the intended or the unintended recipients of a message. In most cases, however, the intended actors will be a subset of the intended recipients. For instance, Alfred Hopkins is the only intended actor in the above example.

- *Actor Awareness Constraint*: An ACL should support a requester's ignorance about the intended actors of the request. For example, a teacher should be able to request

"all those who have done the homework" to raise their hands, without knowing in advance which students have done the homework.

- *Sender's Awareness Constraint*: An ACL should support a sender's ignorance about the individual members of a recipient group. This is typically the case with radio and television broadcasts, notices and banners, and authoring web pages and journal articles.

- *Recipient's Awareness Constraint*: An ACL should support the ignorance of a recipient about other recipients of the same message. The reader of a newspaper article may not know who else read that article, yet she may be able to make certain inferences about the mental state of others who have read or will be reading the same article.

- *Originator Constraint*: An ACL should support a recipient's potential ignorance of the originator or sender of a message. A sign "Authorized Personnel Only" may not indicate the author, but it does communicate the appropriate intentions to anybody who reads the sign. Similarly, a note that I discover on the beach may let me make inferences about the intentions of "whoever wrote the note" even if I don't come to know or deduce its author from it.

We note however, that the FIPA specification [41] supports the actor constraint to some extent.

5.2.2 Adding Scope Rules to Group Notation

We extend the group notation of Section 5.1.1 by introducing scope of the variable in the lambda expression, and further simplify the resulting notation by using the following conventions. We will underline the entities that represent groups when we need to emphasize their group status, and use the same symbol without the underline in a functional notation to denote the associated membership predicate. For example, $\underline{\tau}$ is a group having the membership predicate $\tau(\mathrm{z})$ where z is a free variable. An entity without underline can be either an individual or a group.

With these changes, the notation α denotes a formula defined by the following rule:

1. If α is a formula without any term of the form $\underline{\tau}$, then $\langle\alpha\rangle = \alpha$

2. If α is a formula with term $\underline{\tau}$, and z does not appear in α, and τ(z) is the property predicate that corresponds to $\underline{\tau}$, and α(z) is a formula formed by replacing $\underline{\tau}$ with z in α, then $\langle\alpha\rangle = \forall$z. τ(z) $\supset$ α(z)

For example,

$\langle$BEL x p $\rangle$ = (BEL x p), if x is an individual agent.

$\langle$BEL $\underline{\tau}$ $p\rangle$ = $\forall$z τ(z) $\supset$ (BEL z p)

$\langle$BEL τ $p\rangle$ cannot be further expanded until we know whether τ is an individual or a group.

In case of ambiguity, we will mark the starting angle bracket, and the group term that it applies to, with the free variable in the superscript. This defines the scope of the free variable.

$$\langle^{y}\text{BEL } \underline{\tau}^{y} \text{ (BEL x } \langle^{z}\text{BEL } \underline{\tau}^{z} \ p\rangle)\rangle \quad = \quad \forall\text{y } \tau(y) \supset \text{(BEL y (BEL x } \forall\text{z.}(\tau(\text{z}) \supset \text{(BEL z } p))))$$

If τ represents an individual agent, say x, then the superscript is dropped in the expansion.

$$\langle^{y}\text{BEL } \tau^{y} \ p\rangle = \langle\text{BEL } \tau \ p\rangle = \text{(BEL } \tau \ p) = \text{(BEL x } p)$$

Sometimes, groups need to be treated as meta-agents with agent-like properties and not as a list of individuals. This distinction is discussed in the section on group action. In this case, the membership predicate will not be specified, the term representing this group will not be underlined, and the group will be treated as an individual agent.

5.2.3 Group Beliefs

Our semantics of group communication primitives based on speech acts deals with group beliefs. The simplest case is to consider the beliefs of all the members of a group when talking about group beliefs. The beliefs of more complex groups such as hierarchically composed organizations and institutions [114] can then be expressed in terms of the beliefs of an abstract group consisting of certain roles in that organization or institution.

Group Belief

Group belief may be defined in several ways. This dissertation assumes inclusive belief defined next unless explicitly stated otherwise.

Definition 5.12. Inclusive Belief

A group $\underline{\tau}$ believes p if all the individuals or the sub-groups that constitute the group believe p.

$$\begin{aligned}(\text{BEL } \underline{\tau}\ p) &\triangleq \langle \text{BEL } \underline{\tau}\ p \rangle \\ &= \forall \text{z } \tau(\text{z}) \supset (\text{BEL z } p)\end{aligned}$$

For example, "the students of CSE101 believe p" can be represented by

$$(\text{BEL StudentsOfCSE101 } p) \equiv \forall \text{z (student z CSE101)} \supset (\text{BEL z } p)$$

assuming that the domain membership predicate (student z CSE101) is defined.

Other possible definitions of group belief may include (1) *extensive belief*—mutual belief among all the constituents (individuals or sub-groups) of a group, (2) *existential belief*—belief by at least one constituent of a group, (3) *majority belief*—belief by a majority in a group, and (4) *extensive majority belief*—mutual belief among a majority in a group.

Group Mutual Belief

An entity τ_1 has unilateral mutual belief about a proposition p with another entity τ_2 when τ_1 believes that there is mutual belief between itself and τ_2 about p. It is possible to define different variations of group BMB corresponding to the various types of group beliefs mentioned above. For inclusive beliefs that we assume in this dissertation, we define four different categories of BMB.

Definition 5.13. Unilateral Mutual belief between two individuals
This is the degenerate case in which the two groups happen to be singleton groups. The semantics of (**BMB** x y p) has been given in a previous section. The semantics of all other cases will be expressed in terms of the semantics of this base case.

Definition 5.14. Unilateral Mutual belief between an individual and a group

Agent x has unilateral mutual belief about proposition p with every member of group $\underline{\tau}$ separately. Note that x knows who are the group members.

$$\begin{aligned}(\text{BMB x } \underline{\tau}\ p) &\triangleq \langle \text{BMB x } \underline{\tau}\ p \rangle \\ &\equiv \forall z\ \tau(z) \supset (\text{BMB x z } p)\end{aligned}$$

Definition 5.15. Unilateral Mutual belief between a group and an individual

Every individual in the group $\underline{\tau}$ has unilateral mutual belief about proposition p with agent x.

$$\begin{aligned}(\text{BMB } \underline{\tau} \text{ x } p) &\triangleq \langle \text{BMB } \underline{\tau} \text{ x } p \rangle \\ &\equiv \forall \text{z } \tau(\text{z}) \supset (\text{BMB z x } p)\end{aligned}$$

Definition 5.16. Unilateral Mutual belief between two groups

A group $\underline{\tau_1}$ has unilateral mutual belief about proposition p with another group $\underline{\tau_2}$ when everybody in group $\underline{\tau_1}$ has unilateral mutual belief with every member of group $\underline{\tau_2}$ separately.

$$\begin{aligned}(\text{BMB } \underline{\tau_1}\ \underline{\tau_2}\ p) &\triangleq \langle^{\text{z}}\langle^{w}\text{BMB } \tau_1^{\text{z}}\ \tau_2^{w}\ p\rangle\rangle \\ &\equiv \forall z\ \tau_1(z) \supset\ (\text{BMB z } \underline{\tau_2}\ p)\end{aligned}$$

Definition 5.17. Group Mutual Belief

Given the above definitions of unilateral mutual belief, the entities τ_1 and τ_2 have mutual belief about proposition p when both τ_1 and τ_2 have unilateral mutual beliefs about proposition p with respect to the other entity.

$$(\text{MB } \tau_1\ \tau_2\ p) \triangleq (\text{BMB } \tau_1\ \tau_2\ p) \wedge\ (\text{BMB } \tau_2\ \tau_1\ p)$$

This is a straightforward generalization of the mutual belief defined for two agents (Definition 2.3b).

5.2.4 Group Action

Researchers in multi-agent systems have attempted to answer questions such as what it means for a group to do an action [45]. However, we are mainly interested in the meaning of terms such as (HAPPENS τ a) and (DONE τ a) where a is an action expression and τ is a group.

For the purpose of this dissertation, all we need is to be able to distinguish between (1) a group doing an action as an entity (or meta-agent), and (2) everybody in a list of individuals performing the action. For instance, a request to CSE101 students to move the teacher's desk is a request to the students as a whole. It may entail the CSE101 students deciding which students would do the action of moving the heavy desk and how the individual actions of those students would be coordinated. On the other hand, a

request to everybody in CSE101 to submit the homework is a request to every student in the class to submit their homework individually. An agent communication language should be able to properly convey these nuances of a requester's intentions about the performers of an action. We distinguish between these two cases in our semantics by requiring that the group be treated as a meta-agent in the first case – the membership predicate should not be specified. Terms such as (HAPPENS τ a) do not decompose further and it is a part of the problem solving process of the group to decide how the group does the action a. The second case requires specification of a membership predicate and terms such as (HAPPENS $\underline{\tau}$ a) will be defined as $\langle$HAPPENS $\underline{\tau}$ $a\rangle$. This term expands to $\forall$z τ(z) $\supset$ (HAPPENS z a) requiring every member of the group τ to do the action a.

5.2.5 Group Extension of Basic Concepts

Here we extend the basic semantic concepts from Chapter 2 using the group formulation developed in the previous sections. It is important to note that the definitions to follow allow for both groups and individuals, as τ may either be an individual or a group.

Definition 5.18. Persistent Goal

$$\begin{aligned}(\text{PGOAL } \tau\ p\ q) \triangleq\ & (\text{BEL } \tau\ \neg p) \wedge (\text{GOAL } \tau\ \Diamond p) \wedge \\ & (\text{UNTIL } [(\text{BEL } \tau\ p) \vee (\text{BEL } \tau\ \Box\neg p) \vee (\text{BEL } \tau\ \neg q)](\text{GOAL } \tau\ \Diamond p))\end{aligned}$$

An entity (agent or group) τ having a persistent goal p is committed to that goal. The entity τ cannot give up the goal that p is true in the future, at least until it believes that one of the following is true: p is accomplished, or is impossible, or the relativizing condition q is untrue. The group PGOAL differs from JPG in that everyone in the group has to have a belief about the end state instead of a mutual belief about the end state as in the case of JPG.

Definition 5.19. Intention

(INTEND τ a q) $\triangleq$ (PGOAL τ [HAPPENS τ (BEL τ (HAPPENS a))?;a] q)

Intention to do an action a is a commitment to do the action knowingly. The entity τ is committed to being in a mental state in which it has done the action a and just prior to which it believed that it was about to do the intended action next.

Definition 5.20. Attempt

$$\begin{aligned}(\text{ATTEMPT } \tau\ e\ p\ q\ t) \triangleq\ & t?; [(\text{BEL } \tau\ \neg p) \wedge \\ & (\text{GOAL } \tau\ (\text{HAPPENS } e; \Diamond p?)) \wedge \\ & (\text{INTEND } \tau\ t?; e; q?\ (\text{GOAL } \tau\ (\text{HAPPENS } e; \Diamond p?)))]?; e\end{aligned}$$

An attempt to achieve p via q at time t is a complex action expression in which the entity τ is the actor of event e and just prior to e, the actor chooses that p should eventually become true, and intends that $e$should produce q relative to that choice. So, p represents some ultimate goal that may or may not be achieved by the attempt, while q represents what it takes to make an honest effort.

Definition 5.21. Persistent Weak Achievement Goal

$$\begin{aligned}(\text{PWAG } \tau_1\ \tau_2\ p\ q) \triangleq\ & [\neg(\text{BEL } \tau_1\ p)\ \wedge (\text{PGOAL } \tau_1\ p\ q)]\vee \\ & [(\text{BEL } \tau_1\ p)\ \wedge (\text{PGOAL } \tau_1\ (\text{MB } \tau_1\ \tau_2\ p)\ q)]\vee \\ & [(\text{BEL } \tau_1\ \Box\neg\ p)\ \wedge (\text{PGOAL }\ \tau_1\ (\text{MB } \tau_1\ \tau_2\ \Box\neg p)\ q)]\vee \\ & [(\text{BEL } \tau_1\ \neg q)\ \wedge (\text{PGOAL } \tau_1\ (\text{MB } \tau_1\ \tau_2\ \neg q))]\end{aligned}$$

This definition states that an entity τ_1 has a PWAG with respect to another entity τ_2 when the following holds: (1) if entity τ_1 believes that p is not currently true, it will have a persistent goal to achieve p, (2) if it believes p to be either true, or to be impossible, or if it believes the relativizing condition q to be false, then it will adopt a persistent goal to bring about the corresponding mutual belief with entity τ_2. As we have seen in Chapter 2, PWAG is a basic concept in joint intentions and is used in the definition of request.

A Generalized Communication Primitive

We now present a definition of the request performative with group semantics. This definition is a generalized version of the individual communication performative in Definition 2.13. The terms α, β, and γ in the following definition can represent either groups or individuals. Here, α is the entity performing the request, β is the recipient (including the "over-hearers") of the request message, and γ is the intended actor.

Definition 5.22. Request

(REQUEST $\alpha\ \beta\ \gamma\ e\ a\ q\ t$) $\triangleq$ (ATTEMPT $\alpha\ e\ \phi\ \psi\ t$)

where,

$\phi = \langle^z(\text{DONE } \gamma^z\ a) \wedge [\text{PWAG } \gamma^z\ \alpha\ (\text{DONE } \gamma^z\ a)\ (\text{PWAG } \alpha\ \gamma\ \langle^w\text{DONE } \gamma^w\ a \rangle\ q)]\rangle$
and $\psi = [\text{BMB } \beta\ \alpha\ (\text{BEFORE } e\ [\text{GOAL } \alpha\ (\text{AFTER } e\ [\text{PWAG } \alpha\ \gamma\ \phi\ q]\ )]\ )]$

Substituting for ϕ and ψ in the definition of attempt (Definition 5.20), we get the goal and the intention of the request respectively. The goal of the request is that the intended actor γ eventually does the action a and also has a PWAG with respect to the

requester α to do a. The intended actor's PWAG is with respect to the requester's PWAG (towards her) that she does the action a. The requester's PWAG is itself relative to some higher-level goal q. The intention of the request is that the recipient β believe there is a mutual belief between the recipient and the requester that before sending the request, the requester α had a goal that after sending the request he (the requester) will have a PWAG with respect to the intended actor γ about the goal ϕ of the request. Next, we see that our definition of request never requires a requester to know who the recipients (both intended and unintended) or the intended actors are.

The recipient β and the intended actor γ never quantify into the beliefs of the requester α - meaning thereby that the requester α does not need to know who β and γ are. Let us consider the general case in which β and γ are groups with specified membership predicate and α could be either a group or an individual, that is, consider (REQUEST α $\underline{\beta}$ $\underline{\gamma}$ e a q t). The term $\langle^z$(DONE $\underline{\gamma}^z$ a)$\dots\rangle$ in ϕ expands to ($\forall$z γ(z) $\supset$ $\dots$) with $\underline{\gamma}^z$ replaced by z everywhere. After plugging ϕ into the definition of attempt (Definition 5.20) and simplifying, we get (GOAL α $\dots$($\forall$z γ(z) $\supset$ $\dots$)) which means that the requester does not have to know about the members of the group $\underline{\gamma}$. The PWAG conjunct of ϕ has the requester's PWAG as its relativizing condition (PWAG α $\underline{\gamma}\dots$). However, the $\underline{\gamma}$ in (PWAG α $\underline{\gamma}\dots$) is not specified as $\underline{\gamma}^z$ so it does not get replaced by the z that appears in $\langle^z$(DONE $\underline{\gamma}^z$ a)$\dots\rangle$ and hence $\underline{\gamma}$ does not quantify into the requester's PWAG as a result of expanding the angle brackets in ϕ. From Definition 5.21, the (PWAG α $\underline{\gamma}\dots$) expands to terms of the form [(BEL α p)$\wedge$ (PGOAL α (MB α $\underline{\gamma}$ p))]. Expanding the MB in terms of BMB and between two groups, the only relevant term that we get is of the form (PGOAL α (BMB $\underline{\gamma}$ α p) $\dots$). Using the definition of inclusive BMB given earlier, this expression further simplifies to

(PGOAL α [$\forall$z.γ(z) $\supset$ (BMB z α p))$\dots$..])

where z is a variable that has not been used anywhere else in the expansion of request. Here also, $\underline{\gamma}$ does not quantify into the beliefs of the requester α. It is important to note, however, that any other definition of group BMB (such as exclusive BMB) will also not quantify $\underline{\gamma}$ into the beliefs of the requester α in the term (PGOAL α .....).

By plugging ψ into attempt, and with similar reasoning we find that the term ($\dots$[PWAG α $\underline{\gamma}$ ϕ q]$\dots$) does not quantify the intended recipient $\underline{\gamma}$ into the beliefs and goals of the requester α. Moreover, the term (INTEND$\dots$[BMB $\underline{\beta}$ α]$\dots$) in the expansion of attempt after plugging ψ, never quantifies the recipient $\underline{\beta}$ into the beliefs and goals of the requester α, as can be seen by similar expansion and reasoning. Hence, we see that *our*

definition of request never requires a requester to know who the recipients (both intended and unintended) or the intended actors are.

We now illustrate examples of usage of this request.

Examples of Usage

Example 1. A request from one agent x to another agent y

This is the degenerate case in which each of the communicating groups consists of a single agent. The recipient of the message and the intended actor will be the same agent. Using the rules for expanding our macro notation, the above definition reduces to the following:

(REQUEST x y y e a q t) $\equiv$ (ATTEMPT x e ϕ ψ t)
where ϕ = [(DONE y a) $\wedge$ [PWAG y x (DONE y a) (PWAG x y (DONE y a) q)]]
and ψ = [BMB y x (BEFORE e [GOAL x (AFTER e [PWAG x y ϕ q])])]

As expected, this expression is same as the definition of request between two agents in (Definition 2.13).

Example 2. "All those who have done the homework raise their hands".

Here, the requester α is a single agent – the teacher. The recipient β is a group—all students in the class. The intended actor γ is also a group—all the students in the class who have done their homework. The action a is "raise hand". Formally, this request may be expressed as

(REQUEST teacher
students_in_class
students_done_homework
e
raise_hand
homework_due(now)
t)

Let us assume that the membership predicate for γ ie. students_done_homework is (doneHomework z).

The goal term ϕ in the definition of request expands to the following:

ϕ = $\forall z$.(doneHomework z) $\supset$ [(DONE z raise_hand) $\wedge$
[PWAG z teacher (DONE z raise_hand)
(PWAG teacher students_done_homework
⟨DONE students_done_homework raise_hand⟩
homework_due(now))
]]

The goal part of the request is that every student z that has done the homework eventually does the action of raising her hand. Moreover, the student z should also have a PWAG with respect to the teacher that she (the student z) does the action of raising her hand. Furthermore, this PWAG should be with respect to the teacher's PWAG with "the students who have done their homework" that all students who have done their homework do the action of raising their hands. The intention of the request is to have mutual belief with all students (irrespective of whether or not they have done the homework) in the class about this goal.

Meeting the requirements

What makes this definition of request uniquely powerful is that it satisfies all the constraints on agent communication languages identified earlier. The addressee and the sender constraints are satisfied because α and β can be groups as well as individuals. The recipient constraint is satisfied because β includes all the recipients – intended as well as unintended. The actor constraint is satisfied because we have a separate term γ for the intended actor. The only place where the recipient β is used in the definition of request is in [BMB β α ...]. From the definition of inclusive BMB used in this dissertation, we see that the members of β do not need to know who the other members of β are. Therefore, the recipient's awareness constraint is supported where it is needed. The originator constraint is satisfied because the requester α does not quantify into the beliefs of the recipient β in the term [BMB β α ...]. The most intriguing part of the request definition, however, is that it even satisfies the actor awareness constraint and the sender's awareness constraints as seen by the following theorems.

Theorem 5.3. *A request can be performed even when the requester does not know who the intended actor is. Formally,*

(Done α (REQUEST α β $\underline{\gamma}$ e a q t)) $\wedge$ $\neg\exists$z.(BEL α γ(z))

is satisfiable.

Proof. Construct a possible worlds model that satisfies both the conjuncts. We use the situation in example 2 to construct such a model. Let the real world w_0 be the world just after the request event has taken place. Let w_1 and w_2 be the worlds that are both belief and goal accessible by the teacher. Let the proposition (Done α (REQUEST α β $\underline{\gamma}$ e a q t)) be true in w_1 and w_2. Let y_1 and y_2 be two students who have done their homework and hence are the intended actors of the request. Suppose that w_1 is belief accessible from w_0 by y_1, and w_2 be belief accessible from w_0 by y_2. The proposition (doneHomework y_1) is true in w_1 and (doneHomework y_2) is true in w_2. However, it is not the case that $\exists$z.(BEL teacher (doneHomework z)) because the z in w_1 and w_2 differ. Since w_1 is the only accessible world for y_1 and w_2 is the only accessible world for y_2, y_1 believes that it has done the homework in w_1, and y_2 believes it has done the homework in w_2, because both y_1 and y_2 know that they individually satisfy (doneHomework z). Therefore, it is possible for the teacher to have the goal that whoever has done the homework be able to evaluate the implication ($\forall$z.(doneHomework z) $\supset$ (DONE z raise_hand) $\wedge$ (PWAG z teacher .....)). This is the goal part of the request that we get after plugging ϕ in the attempt. Similarly, using a membership predicate for the class and constructing worlds in which these propositions hold, the intention part of the request can be satisfied. Therefore, (Done α (REQUEST α β $\underline{\gamma}$ e a q t)) $\wedge$ $\neg\exists$ z.(BEL α γ(z)) is satisfiable in this model. □

Theorem 5.4. *A request can be performed even when the requester does not know everyone who will get the message. Formally,*

(Done α (REQUEST α β γ e a q t)) $\wedge$ $\neg\exists$z.(BEL α β(z))

is satisfiable.

Proof. This follows from the proof of the above theorem when a model is constructed to satisfy the intention part of the request. □

5.2.6 Discussion

Although there has been considerable work in agent communication languages [41, 67], and researchers, including us, have investigated group intentions and group action [104, 45], group communication has not been addressed in a principled manner. We believe the present work provides a first step in this direction. We identified a set of requirements for agent communication languages, presented a generalized request performative that can

handle both group and individual communication, and showed that this performative is novel in that it satisfied all the identified constraints.

The current STAPLE interpreter supports a simplified group communication semantics where the group membership is known. Implementation of the full semantics of group communication in terms of "whoever" is left for future work. The simplified group communication is consistent with the dynamic team assumption (Assumption 5.1) because team members in a dynamic but persistent team will eventually know all other team members. As such, we are left with the problem of defining the simplified group communicative acts and to show that these communicative acts do in fact create and discharge persistent and dynamic teams defined earlier in this chapter. These group communicative acts and the results that follow are used in an implemented STAPLE interpreter presented in this dissertation.

5.3 GROUPS IN STAPLE

Groups in STAPLE can either be named groups such as "Yankees" or they can simply be a collection of known individuals such as the set {Bob, Harry, Peter}. The evaluation of the group membership predicate abstracts away from how groups are actually represented and the discussion that follows is independent of any particular representation of groups. As in the previous sections, the predicate (member x τ) is true if x is a member of the group τ at the time of evaluation and false otherwise. Also, recall that we assume the agents in a team eventually know the identity of the other team-mates (Assumption 5.1). A consequence of this assumption is that under steady state if x is a member of group τ then it is mutually believed by the group τ that x is member of that group. For simplicity, STAPLE does not deal with authorized representative and similar semantic primitives and therefore, communicating with a group involves communicating with every member of the "known" group.

Axiom 5.1. *Communicating with a group in STAPLE involves communicating with every member of that group.*

A consequence of this assumption is that communicative acts addressed to a group must be delivered to each member of the group by the underlying infrastructure. Another consequence is that the group communicative acts in STAPLE do not allow the actor (i.e., sender of the message) to be a group. Next, we discuss group beliefs, group commitments, and group communicative acts in the context of STAPLE.

5.3.1 Group beliefs

We will use both formulations of group mutual belief: (MB τ p) represents that the group τ mutually believes that p (Definition 5.2), and (MB τ_1 τ_2 p) denotes that there is mutual belief between entities τ_1 and τ_2 that p where one or both of entities τ_1 and τ_2 can be groups (Definition 5.17). The following useful properties are a direct consequence of these definitions of group mutual belief.

Proposition 5.1. Properties of group mutual belief

a) (member x τ) $\supset$ (MB τ p) $\equiv$ (MB x τ p)

b) (member x τ) $\wedge$ (member y τ) $\wedge$ (MB τ p) $\supset$ (MB x y p)

c) (BEL y (member x τ)) $\wedge$ (BEL y (MB τ p)) $\supset$ (BEL y (BEL x p))

These properties can be generalized to allow making inferences about group mutual belief from the mutual belief about belief of each of its members. For example, we should be able to make the following deduction assuming that τ is the group {x,y,z}:

(MB τ (BEL x p)) $\wedge$ (MB τ (BEL y p)) $\wedge$ (MB τ (BEL z p)) $\supset$ (MB τ p)

We state this property in the following lemma.

Lemma 5.1. *A group mutually believes that every member of the group believes p iff the group mutually believes that p. Formally,*

(MB τ $\langle$BEL λx.(member x τ) $p\rangle$) = (MB τ p)

Proof. : L.H.S.

= (MB τ $\langle$BEL λx.(member x τ) $p\rangle$)
= $\langle$BMB λz.(member z τ) τ $\langle$BEL λx.(member x τ) $p\rangle\rangle$ [From Defn. 5.2 of MB]
= $\langle$BMB λz.(member z τ) τ $p\rangle$ [From defn. 5.1 of (BMB z τ)]
= (MB τ p) [From Defn. 5.2 of MB]
= R.H.S.

□

Next, we analyze the commitments of a group of agents towards each other.

5.3.2 Group PWAG and JPG

The group version of PWAG in Definition 5.21 is built from the group version of its constituent terms. Here we investigate how the PWAG of a group (or towards a group) relates to the PWAG of (or towards) the individual members of the group. These results are useful for establishing and reasoning about JPG in a group of agents and they are encoded as STAPLE reasoning rules.

Lemma 5.2. *If for every member y of a group τ there is mutual belief between an agent x and the group τ that member y has a PWAG towards x then it is mutually believed between them that the group has a PWAG towards x. In other words, PWAG of every member of a group towards an agent implies PWAG of the group as a whole towards that agent using the above definitions of group beliefs and group goal. Formally,*

$$\models \langle \text{MB x } \tau(\text{PWAG } \lambda\text{y.(member y } \tau) \text{ x } p\ q)\rangle \supset (\text{MB x } \tau\ (\text{PWAG } \tau \text{ x } p\ q))$$

Note that the above formulation quantifies the team members y into the beliefs of x and τ.

Proof. We want to show that the LHS leads to the same mental attitude that the RHS is reduced into whenever any disjunct in the definition of PWAG is true. We need to consider four exhaustive cases – one for each disjunct in the definition of PWAG.

Case 1: $\langle\neg(\text{BEL } \tau\ p)\rangle$

In this case,

$\langle \text{MB x } \tau\ (\text{PWAG } \lambda\text{y.(member y } \tau) \text{ x } p\ q)\rangle \wedge \langle\neg(\text{BEL } \tau\ p)\rangle$

$\supset [\forall\text{y (member y } \tau)\ \supset (\text{MB x } \tau\ (\text{PWAG y x } p\ q))] \wedge [\forall\text{z (member z } \tau)\ \supset\ \neg(\text{BEL z } p)]$

$\supset [\forall\text{y (member y } \tau)\ \supset (\text{MB x } \tau\ (\text{PGOAL y } p\ q))]$

$\supset (\text{MB x } \tau\ (\text{PGOAL } \tau\ p\ q))$ [Using definitions group belief and group goal]

Note that the RHS is reduced to (MB x τ (PWAG τ x p q)) from the definition of group PWAG when $\neg$(BEL τ p). The steps for the other three cases (BEL τ p), (BEL τ $\Box\neg p$) and (BEL τ $\neg q$) are similar. This establishes the desired result. □

Lemma 5.3. *If there is mutual belief between a group member x and the group τ that some agent y has a PWAG towards the group then it is mutually believed between that group member and the group that agent y has a PWAG towards group member x.*

$$\models (\text{member x } \tau) \wedge (\text{MB x } \tau\ (\text{PWAG y } \tau\ p\ q)) \supset (\text{MB x } \tau\ (\text{PWAG y x } p\ q))$$

Proof. : As in the above lemma, we want to show that the LHS leads to the same mental attitude that the RHS is reduced into whenever any disjunct in the definition of PWAG is true.

Lets consider the case when (BEL τ p). From the definition of group belief, it follows that (BEL x p). Therefore, the RHS reduces to (MB x τ (PGOAL y (MB y x p) q)) using the second disjunct in the definition of PWAG. From the LHS, using the definition of PWAG, it follows that (MB x τ (PGOAL y (MB y τ p) q)). However, (MB y τ p) $\supset$ (MB y x p) where (member x τ). Therefore, LHS leads to RHS in this case. The steps for the other three cases $\langle\neg$(BEL τ p)$\rangle$, (BEL τ $\Box\neg p$) and (BEL τ $\neg q$) are similar. This establishes the desired result. □

The next theorem generalizes the interlocking PWAG theorem for establishing JPG in a group of agents.

Theorem 5.5. *Mutual belief in a group that every group member has a PWAG towards the group to achieve a goal p is sufficient to establish a joint commitment in the group to achieve p provided that (1) there is mutual belief in the group that p has not already been achieved, and (2) the PWAGs are interlocking, that is, the PWAG of all but one group member towards the group is relative to the PWAG of one of the group members towards the group. Formally,*

$$\begin{aligned} \models [(\text{MB } \tau \text{ (PWAG x } \tau \ p \ q)) \quad &\wedge \quad \langle \text{MB } \tau \text{ (PWAG } \lambda\text{z.(member z } \tau) \ \tau \ p \ r{\wedge}q)\rangle \wedge (\text{MB } \tau \ \neg p)] \\ &\supset \quad (\text{JPG } \tau \ p \ r{\wedge}q) \\ &\quad\ \text{where } r = (\text{PWAG x } \tau \ p \ q), \text{ and (member x } \tau) \text{ is true} \end{aligned}$$

Proof. We need to show that all the three conjuncts in the definition of (JPG τ p $r{\wedge}q$) follow from the antecedent. The proof is similar to that of Theorem 2.1 where the group consists of two agents.

(a) (MB τ $\neg p$) is trivially follows from the antecedent.

(b) To show (MG τ p) follows from the antecedent:

(i) $\forall$z (member z τ) $\supset$ (MB τ (PWAG z τ p $r{\wedge}q$)) [From antecedent]

(ii) $\forall$z (member z τ) $\supset$ (MB z $\neg p$) [From antecedent]

(iii) $\forall$z (member z τ) $\supset$ (MB τ (GOAL z $\Diamond p$)) [From (i), (ii) and definitions of PWAG and PGOAL]

(iv) (MG τ p) [From (iii) using definition of group mutual goal]

(c) Similarly, to establish {UNTIL [(MB τ p) $\vee$ (MB τ $\Box\neg p$) $\vee$ (MB τ $\neg q$)] (WMG τ p q)}, we follow the same steps as in Theorem 2.1.

This establishes the desired result. □

An important consequence of this theorem is that JPG can be established in a group using group communication. The STAPLE versions of group communicative actions are discussed next.

Group Communicative Acts in STAPLE

The definitions of REQUEST and INFORM defined in Table 4.5 remain unchanged even after introducing group terms. The variables for sender and recipient in those definitions when bound to a group term modify the definitions appropriately to support groups. However, these definitions quantify the recipient into the sender's beliefs and therefore, the senders have to know the recipients of a communicative act. As such, these are simplified definitions of group communicative acts and they do not support full-fledged group semantics as discussed in the previous section. The support for complete semantics of group communication in STAPLE is part of future work.

The following results derived from Theorem 2.3 is used in STAPLE for establishing mutual belief in a group using the INFORM communicative act. These results are encoded as STAPLE reasoning rules.

Theorem 5.6. *An INFORM communicative act from an individual to a group establishes mutual belief between the individual (sender) and the group (recipient) that the sender believes the informed proposition. If the group mutual believes that the sender is a member of that group then the aforesaid INFORM establishes mutual belief in the group that the sender believes the informed proposition. Formally,*

$\models$ (DONE (INFORM x τ e p t)) $\Rightarrow$ (MB x τ (BEL x p))
$\models$ (DONE (INFORM x τ e p t)) $\wedge$ (MB τ (member x τ)) $\Rightarrow$ (MB τ (BEL x p))
$\models$ (DONE (INFORM x τ e (BEL x p) t)) $\wedge$ (MB τ (member x τ)) $\Rightarrow$ (MB τ (BEL x p))

*Proof. T*hese results follow from the definition of INFORM by applying Theorem 2.3 and using the definition of group mutual belief. Theorem 2.3 is still applicable because we are treating groups as individuals in STAPLE. □

Lemma 5.4. *A group mutually believes that p when all members of the group have informed the group that they believe p. Formally,*

$\models \forall$z (member z τ) $\supset$ [(DONE (INFORM z τ e (BEL z p) t)) $\wedge$ (MB τ (member z τ))]
$\Rightarrow$ (MB τ (BEL x p))

Proof. When all group members have responded, assuming that on one has changed its mind we have

$\forall$z (member z τ) $\supset$ (MB τ (BEL z p)) [From Theorem 5.6]
= (MB τ p) [From definition of group MB] □

The definition of agree in Table 4.6 is modified to support groups because we need to separate the notion of an addressee from that of a recipient. For convenience, we define an action `gagree(X,G,Y,action(Act,G,EId),Q)` where `X` is the sender, `G` is the addressee group and `Y` is the recipient whose effect is
`mb(X,G,pwag(X,G,done(action(Act,G,EId)),pwag(Y,G,done(action(Act,G,EId)),Q)∧ Q))`

Group AGREE is used in conjunction with the REQUEST communicative in STAPLE to establish joint commitment in a group of agents as stated in the following lemma.

Lemma 5.5. *A REQUEST to a group to do an action followed by group AGREE (gagree) from every member of the group members establishes JPG in the group to do that action.*

$\models$ (MB τ (member x τ))$\wedge$ (DONE (REQUEST x τ a q))
$\wedge$ {$\forall$z (member z τ) $\supset$ (DONE (AGREE z τ (BEL z p)))}
$\supset$ (JPG τ p $r \wedge q$)
where r = (PWAG x τ p q)

Proof. This result follows directly from Theorem 5.5 using definitions of REQUEST and AGREE. □

Next, we summarize this chapter and then use the concepts developed here to specify a fault-tolerant brokering behavior in the next chapter.

5.4 SUMMARY

This chapter extended the teamwork theory and group communication semantics for use by groups. We found that the same basic concepts that underlie group communication form the core of persistent but dynamic teams. We modified the definition of joint commitment to use group as an entity and argued that the resulting teams continue to exist even when the team membership changes. Maintenance goals add another dimension to team

persistence and we proposed a definition of teams based on maintenance goals that allows teams to continue to exist beyond one time achievement goals. We proposed a framework for representing the semantics of group interaction, used to define the semantics of group REQUEST, and showed that it meets several desirable characteristics. Finally, we proposed a simplified version of group communication that is currently used in the STAPLE agent programming language, and showed that these group communicative acts can be used to establish and discharge persistent teams consisting of groups of agents.

The semantics of group communication in terms of "whoever" as presented in this chapter is not yet implemented in STAPLE. The support for this group communication semantics as well as support for group protocols in STAPLE is left for future work. We also believe that treatment of roles and responsibilities in teams, organizations, and institutions is needed for a better understanding of what happens in group-communication in these complex groups and is part of future work.

Chapter 6

Adaptive Agent Architecture

Brokered multi-agent systems can be incapacitated and rendered non-functional when the brokers become inaccessible due to failures that can occur in any distributed software system. We hypothesize that the theory of teamwork can be effectively combined with basic fault-tolerance principles to specify robust brokered architectures that can recover from broker failures. To show the feasibility of this approach, we present the Adaptive Agent Architecture (AAA). In particular, we argue that (1) teamwork can be used to create a robust brokered architecture that will recover a multi-agent system from broker failures without incurring undue overheads, and (2) teamwork can be used to guarantee a specified number of brokers in a large multi-agent system.

The motivation for the AAA is two-fold. First, it shows the applicability of teamwork to one of the mainstream problems in computer science. Second, it provides a test case for evaluation of STAPLE that we propose in the next chapter. Specifically, the fault-tolerance of brokers in AAA is achieved by implementing a joint intention specification of that fault-tolerant behavior. One of the test cases for STAPLE would be to give this logical specification of fault-tolerance to brokers written in STAPLE and obtain the same behavior as that of the AAA brokers *without* having to actually implement that specification

6.1 INTRODUCTION

Multi-agent systems are prone to failures that can occur in any distributed software system. Bugs and improperly handled exceptions in the agent program or in the supporting environment, machine crashes, network partitioning, and numerous other hardware and software faults can make agents unavailable suddenly for unforeseen periods. The traditional distributed systems literature provides various fault-tolerance techniques to recover

from these failures. We have argued in [61] that most of these techniques are meant for specific failure situations and they require special infrastructural support. For example, the techniques of hot backups [6], object group replication [7], virtual synchrony [7], and N-version voting [20] need specific mechanisms for communication and synchronization among the replicas. It may not be possible to use these techniques in multi-agent systems without extensive modifications to the underlying agent infrastructure. On the other hand, a technique based on a multi-agent system concept may be amenable to implementation with minimal modifications, for example, by adding a plan to the plan library of agents, or in the case of STAPLE agents, by simply specifying an appropriate joint action expression. We note that earlier work on teamwork [70] has shown agent teams to be more robust than a collection of agents in the face of adversity and unforeseen situations [52, 111, 55]. The reason behind this robust behavior exhibited by teams is that the members of a team are committed not only to the success of their portion of the joint action but also to the success of the team as a whole. A team will try to recover from problems and will abandon the joint goal only when it is mutually believed by the team members that the goal is no longer possible. This discussion motivates us to investigate exploiting teamwork to achieve fault-tolerance.

Multi-agent systems often require brokers for accepting requests, locating capable agents, routing requests and responses, sharing of information, managing the system, registering agent capabilities, and for various other facilitation tasks [37]. As a result, a large number of multi-agent infrastructures such as Open Agent Architecture (OAA) [72, 23], RETSINA [116], JATLITE [53], and Infosleuth [78] provide some kind of middle agents or at least some form of facilitation and routing service. However, brokered systems are brittle because the broker is a single point of failure. Our experience with Quickset [24], a multi-agent system based on the brokered OAA, reinforces the need for an agent architecture that can recover quickly from broker failures.

As a multi-agent system with brokers becomes more complex and as the number of agents in the system increases, it will typically use more brokers (or middle agents) to achieve an optimum between redundancy, resource utilization, efficiency, and load balancing. Moreover, when a number of independent multi-agent communities are interconnected, it is generally desirable for each local agent community to have its own middle agents. As such, there will typically be more than one middle agent in the system that may be able to substitute for each other when needed. Therefore, if these middle agents form a team with appropriate joint commitments, they will substitute for any middle

agent that becomes unavailable. As a result, the multi-agent system will continue to work as long as there is at least one middle agent remaining in the broker team. However, the performance may degrade as a result of having fewer middle agents in the system.

This teamwork-based recovery scheme can be extended to have at least N brokers in the system at all times. A broker team with a joint commitment to maintain a specified minimum number of brokers in the team will attempt to restore the population of the team when brokers fail or become inaccessible. One way to achieve this end is by starting new brokers in the system. This approach will result in a recovered system that may have similar performance as the original system. However, support is required from the agent library to enable the agents (including the brokers) to start other brokers. Proper coordination is required among the brokers to ensure correct mutual beliefs, to track the progress of a recovery process, and to reorganize the agents after recovery.

The remainder of this chapter is organized as follows. We review the work done in the area of fault tolerance in multi-agent systems as well as traditional distributed systems in the next section, and introduce the Adaptive Agent Architecture in Section 6.3. We show in Section 6.4 that a multi-agent system can recover from broker failures if the brokers form a team with certain commitments and the agent architecture enables the brokers to honor those commitments. In Section 6.5, we show that an AAA-based multi-agent system can maintain a specified number of brokers in the system due to the joint commitment towards a maintenance goal. Finally, we conclude in Section 6.6 with a summary of the AAA experience and set up a test case that brokers in STAPLE must be able to exhibit.

Next, we review the main fault-tolerance techniques used in literature.

6.2 REVIEW OF FAULT TOLERANCE TECHNIQUES

Here we review some of the main approaches to fault-tolerance in multi-agent systems as well as in the traditional distributed and database systems.

6.2.1 Fault Handling in Multi-agent systems

Two divergent approaches that have been used to diagnose failures and attempt recovery are (1) using sentinels external to the agents that monitor inter-agent communication, and (2) using introspection to monitor an agent's own run time behavior.

Jennings showed that as the world becomes more complex and variable and plans tend

to fail more often, teams as a whole waste fewer resources and are more robust than self-interested agents [52]. This approach is similar to ours in that both approaches are based on the theory of teamwork. However, we explicitly address the problem of fault-tolerance whereas Jennings work is more focused towards cooperative problem solving.

Hägg uses external sentinel agents to monitor inter-agent communication, build models of other agents, and take corrective actions [47]. The sentinel agents listen to all broadcast communication, interact with other agents, and use timers to detect agent crashes and communication link failures. A sentinel agent copies the world model of other agents and detects inconsistencies by observing the behavior of other agents as well as its own internal state. In our teamwork-based approach, the problem solving agents themselves participate in fault-tolerance as opposed to the external sentinel agents used in this work. The sentinels in Hägg's approach analyze the entire communication going on in the multi-agent system to detect state inconsistencies. However, this approach is not realistic for systems such as Quickset [24] due to the high volume and frequency of messages in these systems. Furthermore, the sentinel-based approach may not work if the agents do not make their communication and world model publicly accessible.

Klein proposes to use exception-handling service to monitor the overall progress of a multi-agent system [58]. Agents register a model of their normative behavior with the exceptional-handling service that generates sentinels to guard the possible error modes. The exceptional-handling services use a query and action language to interact with the problem solving agents to detect and diagnose faults and take corrective actions. The exception-handling service is a centralized approach whereas our teamwork-based approach is essentially a decentralized approach. Moreover, Klein's approach relies on being able to communicate with the agents whereas the current work attempts to restore connectivity when communication with a broker is not possible.

Kaminka and Tambe use a social diagnosis approach wherein socially similar agents compare their own state with the state of other agents for detecting possible failures [55]. An explicit teamwork model is used for failure diagnosis wherein agents use plan recognition from observable actions as well as communication with other agents to infer and construct a model of the other agents. This work is similar to the current work in that the teamwork model is used in both cases. However, we mainly concentrate on middle agents whereas Kaminka's work is related to a system that is not based on middle agents.

Decker, Sycara, and Williamson advocate the use of caching information (returned by matchmakers) by individual agents in systems that use matchmakers to improve robustness

in the face of matchmaker failures [37]. Their approach could be applied in the AAA by enabling agents to query brokers for agent capabilities and caching that information for directly contacting other agents in the event of broker failures. They have also shown that using load balancing by brokers in brokered systems improves performance and hence provides a degree of robustness from aggressive agents. These approaches compliment our work and they can be used along with our teamwork-based techniques.

6.2.2 Traditional Fault-Tolerance Techniques

A large number of techniques for fault-tolerance can be found in the traditional database and distributed systems literature. Table 6.1 lists some of the techniques that have been developed for database recovery, for application recovery and recovery of distributed systems. We also briefly review some of the important fault-tolerance techniques used in the database and traditional distributed systems literature and observe that redundancy is the basic principle behind most of those methods. Most of these recovery methods pri-

Table 6.1: Traditional Fault-Tolerance Techniques

Database Recovery:
Redo-undo Logs, Fuzzy and Basic Checkpointing, Database Replication
TP Monitors, Application Servers, Resource Managers:
Recoverable Queues, Pseudo-conversations, Fault-tolerant Input Logging, Checkpointing based Recovery, Transaction based Recovery, Stateless Servers, Warm Backups and Hot Backups, Regenerative Processes
Fault-Tolerant Distributed Systems:
Object Group Replication + Virtual Synchrony, Message Logging, N-Version Voting

marily focus on replication techniques that permit critical system data and services to be duplicated as a way to increase reliability [7]. Active replication is also used for processes wherein the inputs are duplicated and the outputs produced are consolidated. Most of the methods used for application recovery advocate either logging the application messages or frequently saving the application state and therefore, require either a database or a recoverable queue [71] The current work attempts to recover a multi-agent system without recovering an inaccessible broker process and so does not require the brokers to save their state to persistent storage.

Considering the specific problem of recovering a multi-agent system from broker failure,

the closest traditional techniques that may be applicable are the techniques of warm and hot backups and the object group replication technique used in conjunction with virtual synchrony. In the warm backup technique, a process is replicated and when the main process goes down, the replica immediately starts recovering to the last known state of the dead process. A hot backup is similar to a warm backup except that the input and output of the main process as well as the replica are synchronized at all times and so the replica can immediately take over without having to first bring itself to the state of the dead process [6].

The object group replication used with virtual synchrony is essentially the same as the hot backup technique except that here objects are replicated instead of active processes [7]. Groups of objects are treated as a single object and all objects in a group receive the same messages in the same sequence. Therefore, if we form a group of similar objects, there is a high probability that the different objects in a group will be in possibly different but correct states. So if one object fails or gets into some unforeseen problem, another object can take over the responsibility of responding to messages. This is the technique typically used to design fault-tolerant multi-ORB applications in the CORBA world [7, 34]. A slightly different technique is that of N-version voting in which N independently developed modules from the same specification run in parallel and the result is decided by voting [20].

The above three categories of fault-tolerance require explicit replication for the purpose of fault-tolerance. These replicas are overheads in the sense that they exist only as backups and perform no useful task. These techniques also require infrastructural support to keep the process replicas synchronized or to implement object groups and virtual synchrony. The technique that we propose uses the brokers that are already present in the system and it uses teamwork to achieve an effect similar to warm backups and object groups plus virtual synchrony. Moreover, generic agents that already have reasoning and planning capabilities may be able to implement a technique based on multi-agent system concepts with minimal support (for example, by adding a plan and the corresponding actions to the plan and action libraries respectively) as opposed to the aforementioned techniques that require specific infrastructural support.

Next, we introduce AAA and its fault-tolerant features.

6.3 OVERVIEW OF THE ADAPTIVE AGENT ARCHITECTURE

The Adaptive Agent Architecture (AAA) is a brokered multi-agent system architecture under development that provides an infrastructure for building fault-tolerant brokered multi-agent systems. It interoperates with the Open Agent Architecture [72, 23] and it is currently used in multi-agent systems, such as Quickset [24], that place heavy demand on inter-agent communication and yet need to provide acceptable real time response.

The AAA agent library has been developed in Java and it provides an agent shell for developing AAA agents. The library also provides a facilitator agent that serves both as a broker and a matchmaker. Henceforth, in this dissertation, we will refer to the AAA facilitator as the AAA broker. The AAA brokers can be interconnected with each other and the agent library supports both facilitated and direct inter-agent communication. The AAA agents advertise their capabilities as well as an address for connection requests with a broker during registration. TCP/IP is used for network transport and the TCP mechanisms and timeouts are used for detection of connection failures. The brokers as well as other agents can dynamically enter and leave AAA-based multi-agent systems. The AAA brokers form a team for the purpose of fault-tolerance and they share knowledge about who is connected to whom with the team members.

The AAA brokers form a persistent broker team, which we call T, when they register with each other. This broker team satisfies the specifications of a *joint intention* (JI) and a *team maintenance goal* (TMtG) for the purpose of fault-tolerance as specified in the following mission statements.

AAA Mission Statement 1: Whenever an agent registers with the broker team, the brokers have a team intention of connecting with that agent, if it ever disconnects from its broker, as long as it remains registered with the team.

AAA Mission Statement 2: The AAA broker team has a team maintenance goal of having at least N brokers in the team at all times where N is specified during the team formation.

The design of the AAA brokers implements the specification of teamwork that follows from the mission statements. Using the mission statements, along with other logical properties of the AAA, we can establish the commitments of the brokers in the team. These commitments result in fault tolerant behavior when the brokers act rationally and take appropriate actions to honor them. Note that the above commitments are in addition

to any brokering commitments that the broker team may have.

Next, we discuss in detail the specific technique of recovering an AAA-based multi-agent system from broker inaccessibility.

6.4 RECOVERY FROM BROKER FAILURE

Here we discuss the teamwork-based technique used by the AAA brokers (AAA facilitators being used as brokers) for automatically recovering a multi-brokered multi-agent system from sudden broker unavailability. The broker under consideration may be inaccessible due to machine crash, network breakdown, or death of the broker process. We also make the simplifying assumption that the brokers in the system are fully connected. We first present the logical characterization of our teamwork model and briefly describe our recovery scheme. Thereafter, we walk through a recovery scenario describing the commitments involved and the actions taken by the agents.

6.4.1 Formal Characterization

Recall that team activity has been explained in terms of the theory of joint intentions. A joint persistent goal (*JPG*) formalizes the notion of joint commitment. The existence of a JPG between groups of agents is a sufficient condition for the formation of a team with respect to that JPG. Two agents have a joint intention (*JI*) to do an action a if they have a JPG to do a while being in a particular mental state. Joint intention can also be defined in terms of team as an entity rather than the individual agents that constitute a team. This extended definition allows for teams whose members may change dynamically.

We argue that the recovery of an AAA-based multi-agent system from broker failures is a consequence of Mission Statement 1 under the dynamic team assumption (Assumption 5.1). The specification of teamwork prescribed by this mission statement is implemented by the AAA brokers. Next, we formally restate Mission Statement 1 of AAA broker team.

AAA Mission Statement 1:

$\models \forall$y [(agent y) $\wedge$ (**DONE** (registered y T)?) $\wedge$ (dynamic T) $\supset$ (**JI** T a(y) (registered y T))]

where,

a(y) = **WHILE** (registered y T) **DO**

[**IF**$\neg$(connected y T) **THEN** (connect y T)]

T = team of brokers

This statement means that whenever an agent registers with the broker team, the brokers have the joint intention of connecting to the agent, if it ever disconnects, as long as it remains registered with the team. It assumes that registration is property of the broker team rather than of a single broker. One of the properties of AAA is that an agent is considered to be registered with the broker team when it registers with any broker teammate. Another logical property of AAA is that when an agent is registered with the broker team then there is mutual belief in the team about this fact. Using the above specification of Mission Statement 1, along with other logical properties of the AAA, we can establish the commitments of the brokers in the team. These commitments result in fault tolerant behavior when the brokers act rationally and take appropriate actions to act on the commitments.

The individual commitments of team members arising as a result of their joint intention have been established in [70] and the joint intention theory has been applied to action sequences in [28]. Applying the properties of joint intention to the action expression $a(\mathrm{y})$, and using similar reasoning as in [70, 28], we establish the following theorems about the commitments of AAA brokers.

The commitments derived from Mission Statement 1 require the brokers in the broker team to locate and connect with any stranded agents that were connected to a broker teammate that is no longer accessible. As a result, the AAA-based agents, which cannot function without a broker, have access to brokers at all times, even when the broker serving them becomes suddenly unavailable. The following theorems follow from Mission Statement 1 and we show how to establish them in the next section.

Theorem 6.1. *Whenever an agent registers with a broker, the broker has a commitment to make this fact mutually believed by the broker team.*

Theorem 6.2. *When an agent unregisters with a broker, the broker has a commitment to make this fact mutually believed by the broker team.*

Theorem 6.3. *When a broker discovers that an agent that is registered with the team is not connected, it has a commitment to make this fact mutually believed.*

Theorem 6.4. *When an agent that is registered with the broker team gets disconnected, the brokers have a team commitment to connect to that agent.*

Theorem 6.5. *When an agent that is registered with the broker team gets disconnected, all the brokers in the broker team have an individual commitment to connect to that agent.*

Theorem 6.6. *When a broker successfully connects to an agent that is registered with the broker team but got disconnected, it has a commitment to bring about mutual belief about this fact.*

Theorem 6.7. *When a broker that was committed to the disconnected agent's being reconnected to the team, learns that the agent has been reconnected to the broker team, it gives up its commitment to connect to that agent.*

Next, we show how to establish these theorems.

6.4.2 Establishing the Formal Properties

We now formally restate and prove Theorem 6.4, which is one of the main theorems that follow from Mission Statement 1.

Theorem 6.4. This theorem says that when an agent that is registered with the broker team gets disconnected, the brokers have a team commitment to connect to that agent. It can be restated formally in terms of team joint persistent goal.

$\models \forall$y [(agent y) $\wedge$ (MB T (registered y T)) $\wedge$ (MB T $\neg$(connected y T)) $\supset$ (JPG T (DONE (connect y T)) q]

where, q is some escape condition.

Proof. Assuming that the antecedent is true, we want to show that the consequent follows from the AAA mission statement 1. Expanding mission statement 1 using the definition of JI for a group of agents, we get the following team JPG.

ξ = [JPG T {DONE γ?;a(y)} (registered y T)] where,

γ = [MB T (HAPPENS T a(y))], and

a(y) = (WHILE (registered y T) DO [if $\neg$(connected y T) THEN (connect y T)])

Let, a(y) = d(y);(registered y T)?;[$\neg$(connected y T)?;(connect y T)|(connected y T)?];e(y)

where, d(y) denotes the previous iterations, and

e(y) denotes the remaining iterations.

Substituting for a(y), we get

ξ = [JPG T {DONE γ?;d(y);(registered y T)?;$\neg$(connected y T)?;(connect y T);e(y)}
(registered y T)]

We are interested in the iteration in which the antecedent becomes true. Without loss of generality, assume that the action subsequence γ?;d(y);(registered y T)?;$\neg$(connected

y T)? has just been done. Using similar lines of reasoning to Theorem 2 in [28], we can establish the following lemma.

Lemma 6.1. *Joint commitment to an action sequence*

$$\models (\text{JPG T (DONE } a;b)) \wedge (\text{MB T (DONE } a)) \wedge (\text{MB T } \neg(\text{DONE } b))$$
$$\supset (\text{JPG T (DONE } b) \text{ [JPG T (DONE } a;b)])$$

Applying this lemma to a(y) in the mission statement, we get

$\xi \supset \chi$ where,

$\chi =$ [JPG T {DONE (connect y T); e(y)} ξ]

By assumption, (MB T¬(connected y T)) is true. It follows that the present members of the broker team mutually believe that the action ¬(connected y T)? has been done and so they know when the next action (connect y T) is to start. Hence, the brokers in the team now have a team commitment to do (connect y T) with respect to the rest of the overall team commitment.

$\therefore \chi \supset$ (JPG T [DONE (connect y T)] χ)

This proves the desired result.

□

The other theorems that follow from Mission Statement 1 can be proved similarly. Next, we use the theorems in the previous section to explain a typical recovery process.

6.4.3 Recovery Scheme

The recovery scheme that follows from the above commitments can be summarized as follows:

1. Each facilitator joins a recoverable broker team consisting of the adjacent brokers. The entire multi-agent system may consist of such teams with overlapping members.

2. Normally, an agent contacts some broker on startup. The broker to be contacted on startup may be specified by the person (or another agent) starting this agent.

3. Each broker informs the following properties to its adjacent brokers:

   (a) The address for direct connection (lets call it *dc-addr*) of agents that register with it.

   (b) The address of the agent that unregisters with it.

4. Agents listen for request from brokers at address *dc-addr* and they inform a broker about this address at the time of their registration with that broker. Brokers can initiate connection with agents for which they know the direct connection address.

5. When a broker disconnects from its teammates, all the brokers on its team attempt to directly contact the agents that were registered with the now disconnected broker. Contacting an agent involves requesting the agent to connect again with a member of the broker team (the requesting broker).

6. After successfully contacting an agent in this manner, a broker informs this fact to its teammates. The other brokers give up their attempts to contact this agent directly.

The multi-agent system has recovered from failure of the disconnected broker when all the agents registered with that broker have been contacted in this manner. The requests that were in progress at the time of the failure, and hence could not be completed, may be sent again by the requesting agent.

6.4.4 A Recovery Scenario

To demonstrate the fault-tolerant behavior of AAA brokers, we set up a scenario consisting of three brokers and two agents. The brokers are interconnected and each agent is connected to one of the brokers. Figure 6.1 illustrates the initial system setup. A client agent periodically sends requests for which the distance agent is the only capable agent. The three brokers form a robust team as described earlier. This system can function only if both the client and the distance agents are registered with a broker.

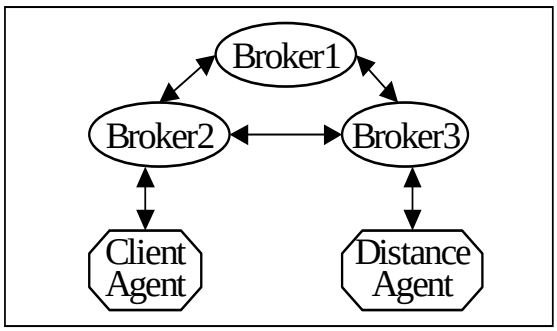


Figure 6.1: Initial Setup

Figure 6.1 illustrates a multi-agent system with three brokers and two other agents.

The client agent periodically sends requests for which the distance agent is the only capable agent. The three brokers form a robust team characterized by the mission statements 1 and 2. This system can function only if both the client and the distance agents are registered with a broker. It has been specified that this multi-agent system should have at least three brokers at all times.

From Theorem 6.1, the brokers have an individual commitment to bring about mutual belief when an agent registers with a broker. Therefore, when the client agent registers with Broker2, Broker2 informs this fact along with the name and address of the distance agent to Broker1 and Broker3. Similarly, when the distance agent registers with Broker3, Broker3 informs this fact to Broker1 and Broker2.

After some time, we kill Broker3. When a broker teammate is no longer accessible, the other brokers believe that all the agents registered with that broker are disconnected (This belief results from an inference rule used by the AAA brokers). When Broker3 is killed, the underlying TCP/IP functionality results in at least one of the remaining brokers discovering that Broker3 is no longer accessible to it and hence, believing that the distance agent is not connected to the broker team. Therefore, from Theorem 6.3, this broker has an individual commitment to bring about a mutual belief about its discovery. As a result, communication is predicted to take place among the brokers in the team.

From Theorem 6.4, the broker team has a joint commitment to connect to the agent that it mutually believes is disconnected from the team. Moreover, from Theorem 6.5, each of the remaining brokers has an individual commitment to contact the disconnected agent. The two brokers act rationally by attempting to contact the distance agent at the address given to them earlier by Broker3. If the distance agent accepts registration request from one of the brokers, it refuses subsequent registration requests from other brokers. Figure 6.2 illustrates the situation when broker1 has successfully contacted the distance agent.

From Theorem 6.6, Broker1 now has an individual commitment to inform the successful connection of the distance agent to its teammates. As a result, Broker1 will act rationally by communicating this information to Broker2, and from Theorem 6.7, Broker2 will give up its attempt if it was still trying to contact the distance agent because the mutual goal has already been achieved. Moreover, from Theorem 6.1, Broker1 needs to inform Broker2 of the registration and address information of the distance agent. In the current implementation of AAA brokers, these two communication attempts are combined into one and just one message is sent from Broker1 to Broker2.

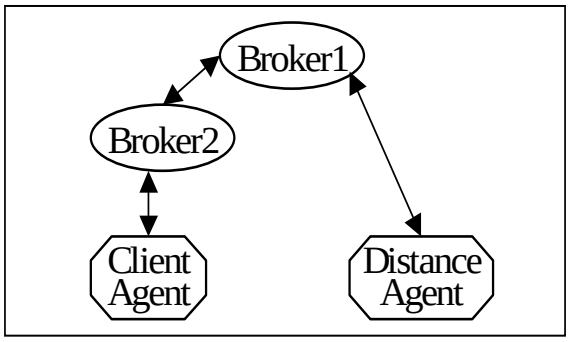


Figure 6.2: After Recovery

After successfully contacting the distance agent, Broker1 requests it for agent-specific information such as its capabilities. Any pending request from the client agent that could not be completed due to the failure of Broker3 may be sent again to the distance agent and the system continues to work.

Next, we discuss the fault-tolerance behavior that arises out of implementing the specification of team maintenance commitment.

6.5 MAINTAINING A SPECIFIED NUMBER OF BROKERS

The current implementation of AAA brokers can maintain at least a specified number of brokers, say N, at all times despite broker failures. The AAA agents commit to the AAA broker, with which they are registered, to honor its requests for starting broker processes. Whenever an agent registers or a broker teammate disconnects from an AAA broker, it checks to see if there are at least N brokers in the team. If not, it searches for an AAA agent on a different machine. If it finds such an agent, it requests that agent to start a broker and request the newly started broker to join the broker team of which the requesting broker is a member[1]. If the AAA broker fails to find an agent on a different machine, it picks up an agent on the local machine at random and repeats the process. Further, an AAA broker started as a result of this process is committed to connect to (and form a team with) the broker that initiated its birth. The AAA broker also keeps track

[1]Note that this algorithm will result in a maximally connected broker graph if (1) the brokers are started one at a time, and (2) a new broker is started after a previously started broker has joined the broker team.

of pending requests to start brokers and may request another agent to start a broker if needed.

This technique resembles the technique of regenerative processes in the traditional fault-tolerance literature wherein a critical process can be restarted by a monitoring process upon failure. However, there are a few major differences between the two techniques.

(1) If the monitoring process fails, there needs to be another level of monitoring process to restart the first monitoring process. This can go up to any level but all these levels have to be explicitly designed and configured for each machine. In the AAA scheme, no separate configuration is needed for each machine. Moreover, all of the requisite N-1 brokers can be started even if there is just one broker left in the system, thereby automatically providing N-1 levels of monitoring.

(2) Special monitoring infrastructure is required to be able to start processes on different machines. A convenient way would be to have separate monitoring processes on each machine that are coordinated using a special distributed algorithm. However, any such algorithm needs a proof of correctness before it can be relied upon. In the AAA scheme, no separate monitoring infrastructure is required as the problem solving agents themselves participate in the fault-tolerance process. The specification of teamwork provides a distributed coordination protocol that we logically prove to work in Section 6.5.2.

6.5.1 Formal Characterization

We now formally restate Mission Statement 2 of the broker team in terms of team maintenance commitment.

AAA Mission Statement 2:

$\models$ (TMtG T [(numberOfBrokers T) >= N])

This mission statement requires the broker team to always have at least a certain number of brokers in the team despite broker failures. The broker team achieves this by getting new brokers started, on possibly different machines, and recruiting them into the broker team. AAA agents can start additional brokers when requested by the broker that is serving them. The end result is that new brokers get started on different machines on the network and the multi-agent system continues to function. The following theorems follow from Mission Statement 2.

Theorem 6.8. *When a broker believes that the number of brokers in the broker team is less than the required number of brokers, and believes that it is not already mutually believed*

and it is not impossible to establish that mutual belief, then it will have an individual commitment to bring about mutual belief of this fact.

Theorem 6.9. *When a broker team mutually believes that the number of brokers in the team is less than the required number of brokers, it adopts a team commitment to have the required number of brokers in the broker team.*

Theorem 6.10. *When a broker team mutually believes that the number of brokers in the team is less than the required number of brokers, the brokers in the team adopt individual commitments to have the required number of brokers in the broker team.*

Theorem 6.11. *When a broker successfully recruits a new broker into the broker team, it has an individual commitment to make this fact mutually believed by the broker team.*

Next, we show how to prove these theorems.

6.5.2 Establishing the Formal Properties

We now formally restate AAA Mission Statement 2 in terms of team persistent goal and prove Theorem 6.8. The other theorems that follow from Mission Statement 2 can be proved on similar lines.

Theorem 6.8. This theorem can be restated formally in terms of team maintenance goal.

$\models$ (TMtG τ p q) $\supset$ {$\forall$x (member x τ) $\supset$
[(BEL x $\neg p \wedge \neg$(MB τ $\neg p$)) $\wedge$ $\neg$(BEL x $\Box\neg$(MB τ $\neg p$)) $\supset$
(PGOAL x (MB τ $\neg p$) (BEL x $\neg p$))]}
where, p is ((numberOfBrokers T) >= N)

Proof. : Let x be a member of team τ and assume that the antecedent (BEL x $\neg p \wedge \neg$(MB τ $\neg p$)) $\wedge$ $\neg$(BEL x $\Box\neg$(MB τ $\neg p$)) is true. From the implication [(BEL x $\neg p$) $\wedge$ $\neg$(MB τ $\neg p$) $\supset$ (GOAL x $\Diamond$(MB τ $\neg p$))] in the definition of WTMtG, we see that (GOAL x $\Diamond$(MB τ $\neg p$)) is satisfied because if one of the team members does not believe there is mutual belief, there is no mutual belief either. Since the consequent of an implication must remain true at least until the antecedent or the implication itself becomes invalid, (GOAL x $\Diamond$(MB τ $\neg p$)) persists until [$\neg$(BEL x $\neg p$) $\vee$ (MB τ $\neg p$) $\vee$ (BEL x $\Box\neg$(MB τ $\neg p$))] becomes true. Hence, all conjuncts in the definition of (PGOAL x (MB τ $\neg p$) (BEL x $\neg p$)) are satisfied. This proves the desired result. $\Box$

Next, we use the theorems from the previous section to explain a typical recovery scenario due to Mission Statement 2.

6.5.3 A Recovery Scenario

Recall that the broker team had a team maintenance goal of having three brokers in the system. Therefore, when a surviving broker discovers that Broker3 is no longer accessible, Theorem 6.8 predicts that it will have a commitment to bring about mutual belief about this fact, and hence, communication will be attempted between the surviving brokers. From Theorem 6.9, the broker team has a team commitment to recruit a new broker to the team and from Theorem 6.10, each of the surviving brokers adopts an individual commitment to have three brokers in the broker team. Broker2 knows of an agent (the client agent) that can spawn a new broker and requests it to do so. After the new broker comes up, it contacts the broker on whose request it was started. Theorem 6.11 predicts that Broker2 will now have an individual commitment to bring about a mutual belief that the broker team has the required number of brokers and therefore, communication will take place among the brokers. At the end of this process, a configuration similar to that in Figure 6.1 will have been restored and hence the multi-agent system has recovered from broker failure.

6.6 CONCLUSION

To summarize, in a large multi-agent system with multiple middle agents the middle agents could potentially serve as backups for each other, thus achieving a high level of fault-tolerance. However, instead of using warm and hot backups, N-version voting and other traditional techniques that build upon explicit redundancy, we propose a scheme that uses teamwork to exploit this inherent redundancy of middle agents to achieve opportunistic service replication. In other words, the recovery scheme based on teamwork avoids the overhead of using redundant brokers just for the purpose of fault-tolerance. We presented the Adaptive Agent Architecture (AAA), a robust brokered architecture, in support of our hypothesis. The AAA is motivated by the theory of teamwork to: (1) recover a multi-agent system broker failures, and (2) maintain a specified minimum number of functional brokers in the system even when some of the brokers become inaccessible.

However, the teamwork based techniques will be most useful when the team specification can be modified easily to get change the team behavior or to introduce a new behavior

without having to implement the modified/new specification by hand as was done in AAA. Recall that STAPLE is an agent programming language where agent and team behaviors can be declaratively specified to get the desired behavior without having to implement those behaviors by hand. If the premise of this chapter is true, we should in principle, be able to give the AAA Mission Statements to brokers written in STAPLE and these brokers should exhibit the same fault-tolerance behavior as that exhibited by AAA brokers. We will see in the next chapter that this is in fact the case and the fault-tolerance behavior of AAA agents due to Mission Statement 1 can be programmed in STAPLE using a few logical sentences.

Chapter 7

Implementing Fault-Tolerance of AAA Brokers in STAPLE

One of the hypotheses for this dissertation is the ability to reproduce the fault-tolerance behavior of AAA brokers in STAPLE just by providing high level specification of that behavior in a few logical sentences. The present chapter seeks to validate this hypothesis by implementing brokers in STAPLE that demonstrate the same fault-tolerance behavior as that of AAA brokers. We will see that it takes only a fraction of programming to implement the AAA fault-tolerant brokers in STAPLE. We will also explore how slightly modifying logical sentences leads to different behavior by agents as well as by broker team. The next section describes implementation of brokers in STAPLE. Section 7.2 discusses the fault-tolerant behavior of these brokers that follows from "Mission Statement 1" of AAA brokers from the previous chapter. Finally, we conclude in Section 7.3 with a brief summary.

7.1 IMPLEMENTING BROKERING BEHAVIOR

STAPLE as a language does not define any brokering behavior or any broker agent. So we need to first implement brokers in STAPLE before we can use those brokers to demonstrate the AAA type fault-tolerance.

7.1.1 Implementing a Broker

A broker is just like any other agent. If the commitments, intentions, and capabilities of an agent lead to a behavior that we understand as brokering behavior then we refer to that agent as a broker. The AAA brokers were implemented as AAA agents that had specialized brokering behavior. It took more than 2500 lines of Java code to implement

the brokering functionality (along with support for fault-tolerance) in an AAA agent. The advantage of a language like STAPLE becomes apparent when trying to implement a broker wherein a broker can be implemented in only two logical sentences. This can be done in STAPLE by simply defining a register action in order to register agents, and a brokering action that results in the broker adopting a commitment for getting the brokered action done. These actions are defined in Table 7.1.

Table 7.1: Implementing a Broker in STAPLE

```
% Action to register an agent

action_definition(register,2) :-
  [args:  [agent(Agent,Host,Port),CapabilityList],
  code:  {reduce_assert(agent_address(Agent,Host,Port)),
          for(Capability, CapabilityList,
                  reduce_assert(can_do(Capability,Agent))
        },
  effects:  [(registered(Agent,self),1.0)]
  ].
% Action to ‘‘broker’’ the action requested by a registered agent

action_definition(broker_action,1) :-
  [args:  [Action],
  code:  {subgoal(pgoal(done(Action)), StackId)},
  stack_id:  StackId,
  effects:  [(brokered(Action),1.0)]
].
```

The register action defined in a broker program calls the belief base maintenance system (`via reduce_assert`) to add the address of the agent, who is registering with this broker, to the broker's belief base. Similarly, it adds each capability of the agent to its belief base using the `can_do/2` predicate. Recall that this STAPLE predicate is used by the rule of rational action to find agents capable of doing a given action. The brokering action (`broker_action`) takes as an argument the action that this broker has been requested to "broker" and it results in the broker having a commitment to get that action done. So when an agent requests a broker agent to "broker" an action and if the broker agent agrees to do that action, it will end up having a commitment to get that action done. From this point onwards, the broker will act on its individual commitment just like any other STAPLE agent by searching for ways to get that committed action done. If the broker can do the action itself, then it will intend and do that action. However, this will

usually not be the case and the broker will look for an agent who is capable of doing that action. If it finds such an agent (say, another agent has registered with this broker and can do the action requested by the first agent) then it will request that agent to do this action similar to the example of using Request communicative action in Chapter 3. We walk through a complete scenario of a brokering behavior in Section 7.1.3.

One advantage of STAPLE is shown by the fact that just adding the above two action definitions to any STAPLE agent program will make that agent behave as a broker! Next, we need to add the corresponding functionality to other STAPLE agents so that they can register with brokers and make use of the brokering service when needed.

7.1.2 Changes to Agents

The STAPLE rule of rational action specifies that if an agent who is committed to achieving p (or for getting an action a done) knows of actions that can achieve p (or get a done) then it will intend to do a non-deterministic OR of those actions. As an example, if an agent x has a commitment to find the distance between two given cities and if it knows of an agent y that is capable of doing the action `find_distance` then one of the actions in the OR expression that agent x intends as a result of the rule of rational action will be a request to agent y do the action `find_distance.`

In order to support usage of brokers, we need to modify the rule of rational action to include the action `broker_action` as a way of getting an action done. More specifically, if an agent is registered with a broker and if it has a commitment to get an action a done then requesting the broker to do the action `broker_action(`a`)` is another possible way of getting the action a done. In this case, the non-deterministic OR action expression that the agent intends as a result of the rule of rational action will include one or more `broker_action` as choices. Alternatively, we can add another rule having a lower importance than the rule of rational action that says that if an agent who is committed to getting an action a done cannot do the action itself and if it does not know of any other agent who can do that action then and if it is registered with a broker then it will request that broker agent to "broker" the action a, i.e, to do the action `broker_action(`a`)`. Giving this new rule a lower importance than the main rule of rational action ensures that this rule fires only when the main rule of rational action does not find any actions that the agent can intend in order to achieve its committed goal(s). Both these options will work for us and we will use the second option (adding a second rule with lower importance) for the examples in the remainder of this chapter. This rule is shown in Table 7.2.

The rule in Table 7.2 says that if an agent has a PGOAL that action `A` be done relative to `Context` and with an importance `Imp`, and if the agent believes that it is registered with a broker agent `Broker` then if it finds a non-empty set of actions that can result in eventually getting the broker to do the action `broker_action(A)` then it will intend a non-deterministic OR of those actions. The `findall` predicate finds a list of actions that this agent believes that it can do, that is, `bel(self,can_do(Action,self))` such that each `Action` can achieve a proposition P which says that eventually the broker agent brokers the action `A`, that is, the broker agent does the action `broker_action(A)`.

Table 7.2: Modifying STAPLE agents to use brokers

```
% Rule to use a broker that this agent is registered with

rule(rational2, pgoal(self, done(A), Context, Imp), 2, StackId) :-
    istrue(bel(self, registered_with_broker(Broker))),
    P = <>(done(action(broker_action(A),Broker))),
    findall(Action,(istrue(bel(self,can_achieve(P, Action))),
                    istrue(bel(self,can_do(Action,self)))),Actions0),
    setof(Actions0,Actions),
    \+ Actions = [],
    NewCtxt = pgoal(self, done(A), Context, Imp),
    NewImp is Imp + 5
    ==>
    subgoal(pgoal(self, done(self,or(Actions)),NewCtxt,NewImp),StackId).

% Rule to register with a given Broker

action_definition(register_with_broker,1) :-
    [args:  [Broker],
    precondition:  bel(self,~intend(self,register_with_broker(_),_Q)),
    code:  {istrue(bel(self,agent_address(Broker,_Host,_Port))),
            findall(Capability,bel(self,capability(Capability)),
                               Capabilities),
            istrue(bel(self,agent_address(self,SHost,SPort))),
            A = action(register(agent(self,SHost,SPort),Capabilities)),
            subgoal(intend(request(self,Broker,A,true)),StackId)
            },
    stack_id:  StackId,
    effects:  [(registered_with_broker(Broker),1.0)]
].
```

In order to use the above rule successfully, an agent must be registered with a broker. We provide the agents with a `register_with_broker` action as defined in Table 7.2. This action takes a broker as an argument, makes sure that the agent knows the address of that broker, creates a list of capabilities of the agent, and then results in the agent having an

intention to request the broker to register it, that is, to do the register action. The broker executes the register action if it agrees to the agent's request to do that action. Successful execution of the register action by the broker registers this agent with the broker and also results in the broker sending a confirmation to the agent that the register action has been done. The register action used by the broker was discussed earlier and is defined in Table 7.1. Also, the creation of the list of capabilities requires that the agent have beliefs about its capabilities `(capability/1)` in its belief base. The STAPLE interpreter could have created a list of capabilities from the actions that are defined for an agent but the agent may not choose to expose all the actions that it knows how to execute as its capability. So we choose to explicitly specify the capabilities of STAPLE agents.

7.1.3 Walking Through a Brokering Example

Recall from the AAA fault tolerance example that we had two agents – a distance agent that could answer questions about distance between cities and a client agent that was interested in finding out distance between cities. Table 7.3 shows the distance agent and Table 7.4 shows the client agent written in STAPLE.

Client and Distance agents

The distance agent has beliefs and intention to register with broker1 and it chooses to register its capability to do the action `find_distance/3`. The action `find_distance/3` is defined to simply get the distance between two cities from the agent's belief base. The agent has a long list of predicates in its belief base that specify the distance between two cities. The action `find_distance` uses this list of to answer question about distance between cities.

The client agent has beliefs and intention to register with broker1 similar to the distance agent. This agent also has another intention to do an action sequence where the first action is to wait until the agent registers with the broker followed by repeatedly getting a city pair via a test action and asking distance between those two cities. The action `ask_distance`[1] is defined such that it results in the agent's committing to do an action `find_distance` that it does not know how to do. The test for city pairs binds a different city pair to the variables `C1` and `C2` in the test action `bel(self,city_pair(C1,C2))?` in each repetition.

[1]Note that the action ask_distance can be alternatively defined using ask-ref and inform-ref that were discussed in the lights world examples. The present example pre-dates the implementation of ask-ref and inform-ref communicative actions in STAPLE.

Table 7.3: Distance agent in STAPLE

```
% agent meta-information
agent_name(self, distance_agent).
agent_version(demo, 2.0, beta).
agent_address(self, localhost, 5045).

% agent's bel, goals, intentions
bel(self, broker(broker1)).
bel(self, agent_address(broker1,localhost, 9045)).
intend(self, register_with_broker(broker1),true,10.1).

% Capabilities that this agent chooses to advertise
bel(self,capability(find_distance(City1,City2,Distance))).

% actions, plans, etc.  specific to this agent
action_definition(find_distance,3) :-
      [args:  [City1,City2,Distance],
      code:  {istrue(bel(self,distance(City1,City2,Distance)))},
      effects:  [(found_distance(City1,City2,Distance),1.0)]
      ].
% Prolog code, data, etc.  for use by this agent's actions, plans, etc.
distance(albany,albuquerque,miles(2040)).
distance(albany,atlanta,miles(1010)).
distance(albany,baltimore,miles(340)).
...
```

Table 7.4: Client agent in STAPLE

```
% agent meta-information
agent_name(self, info_agent).
agent_version(demo, 2.0, beta).
agent_address(self, localhost, 5055).

% agent's bel, goals, intentions
bel(self, broker(broker1)).
bel(self, agent_address(broker1, localhost, 9045)).

intend(self, register_with_broker(broker1), true, 10.0).
intend(self, (wait_for(registered_with_broker(_Broker)),
        (bel(self,city_pair(C1,C2))?,ask_distance(C1,C2))..), true, 5.0).

% actions, plans, etc.  specific to this agent
action_definition(ask_distance,2) :-
        [args:  [City1,City2],
        code:  {subgoal(pgoal(done(find_distance(City1,City2,Distance))),
                        StackId)},
        stack_id:  StackId,
        effects:  [(bel(self,distance(City1,City2,Distance)),1.0)]
        ].

% Prolog code, data, etc.  for use by this agent's actions, plans, etc.
city_pair(C1,C2) :-
        retract(cityindex(N0)),
        N1 is N0+1,
        max_cityindex(Max),
        (N1>Max ->N2=1; N2=N1),
        assert(cityindex(N2)),
        city_pair(N2,C1,C2).

cityindex(0).
max_cityindex(102).

city_pair(1,albany,albuquerque).
city_pair(2,albany,atlanta).
city_pair(3,albany,baltimore).
...
```

We also add the rule and action in Table 7.2 to both distance and client agents to enable them to register with a broker and use the brokering service. Note that the client agent in the above example does not know about the distance agent, it cannot do the action `find_distance,` and it does not know of any agent capable of doing the action find distance. However, if we modify the client agent so that it knows about the distance agent (its name, address, and capability) then instead of going through the broker, it will directly request the distance agent to do action `find_distance.` In this case, the communication that follows will be exactly same as that in the lights world examples.

Demonstrating interactions through a broker

We now illustrate the various interactions that occur when client agent and distance agent communicate through a broker.

(a) Distance agent requests to be registered with the broker

We first start a broker agent named broker1 and then we start the distance agent. The distance agent immediately adopts an intention to do the action `register_with_broker (broker1)` and executes that action as defined in Table 7.2. As a result of executing this action, the agent has an intention to request the broker to do action `register(agent (self,SHost,SPort),Capabilities)` where `SHost` and `SPort` are bound to this agent's address and Capabilities is bound to this agent's list of capabilities which happens to be a list containing only one term `find_distance(City1,City2,Distance).` This intention is with respect to the earlier intention and it is pushed on the same stack on top of that intention. As in the lights world example, the communication that follows is exactly the same as what happens when one agent requests another agent to do an action. The agent pops the intention to do the request action off the stack as it succeeded and replaces it with the commitment that follows. Recall that this commitment is the PWAG that follows from the definition of request. Meanwhile, the broker analyzes the request the action, asserts in its belief base that it is mutually believed that the distance agent has a PWAG that the broker does the action of registering the distance agent relative to distance agent's PWAG with respect to the broker to do that action. The broker decides to adopt that PWAG and therefore, it adopts a commitment to establish mutual belief with the distance agent that it will have this PWAG. This commitment results in the broker intending to perform an AGREE communicative act. It performs the AGREE (that results in an agree message

sent to distance agent) and adopts the resulting PWAG. As a result of this PWAG, the broker has an individual commitment to perform the register action that was requested by the distance agent after which it intends to do the register action. Meanwhile, the AGREE communicative act received by the distance agent is analyzed and the resulting mutual belief is asserted into the distance agent's belief base. This mutual belief discharges the commitment of the distance agent that the broker have a PWAG towards the distance agent for registering it (this is one of the commitments that follows from its PWAG as a result of performing the REQUEST communicative act).

(b) Broker registers the distance agent and establishes mutual belief about this fact

The broker acts on its intention to register the distance agent by executing the `register` action defined in Table 7.1. After discharging this intention and the commitment that lead to this intention, the broker re-evaluates its PWAG that resulted from its performing the AGREE communicative act. It results in the broker's having a PGOAL to establish mutual belief with the distance agent that the register action requested by it just got done. This PGOAL results in an intention to perform an OR expression of actions each of which can establish that mutual belief. One of the actions in the OR expression is a plan to establish mutual belief that all STAPLE agents have access to and it is shown in Table 7.5. The action expression of this plan consists of the agent first establishing a mutual belief that it believes P followed by the agent testing whether it is mutually believed if that this agent is competent with respect to the informed proposition. If such a mutual belief exists, then just establishing the mutual belief that this agent believes P establishes the mutual belief that P, otherwise the agent waits for the other agent to establish mutual belief that it believes the informed proposition P.

In this case, the proposition P is (DONE register(Agent,CapabilityList)) and there is no mutual belief between the distance agent and the broker that the broker is competent with respect this proposition. The broker commits to the first action in the action sequence that is the body of the plan and intends an OR expression to establish mutual belief that `bel(self,done(...))`. The OR expression consists of alternative actions and plans each of which can potentially establish the required mutual belief. This time the agent cannot intend the same plan to establish this mutual belief because of the precondition of this plan so it eventually ends up intending to perform an INFORM communicative act that is

Table 7.5: Plan to Establish Mutual Belief

```
plan(establish_mb,2) :-
    [description:  'Plan to establish mutual belief',
    args:  [Other,P],
    precondition:  {\+ P=bel(self,P1)}, %To avoid infinite loop
    effects:  [(mb(self,Other,P),1.0)],
    body:  {achieve(mb(self,Other,bel(self,P))),
            (((~bel(self,mb(self,Other,competent(self,P))))?,
              wait_for(mb(self,Other,bel(Other,P))))
            ;
            bel(self,mb(self,Other,competent(self,P)))?
            )
           },
     ].
```

also part of the OR expression. It performs the INFORM[2] and discharges the intention to perform this communicative act along with the intention to perform the OR expression. It then commits to the next part of the action sequence which is another OR expression and then commits to one of the actions in the OR expression. Irrespective of which OR branch it chooses first, it ends up discovering that there is no mutual belief about its competence with respect to the proposition that it wants to be mutually believed and so ends up having a commitment to wait for the distance agent to establish mutual belief that it also believes the informed proposition (that the register action is done).

The distance agent gets the INFORM message, analyzes it, asserts the resulting mutual belief that the broker believes that it has done the informed proposition. It infers that the broker is trying to establish mutual belief and it checks that it does not have any private beliefs to the contrary. As such, it tries to be cooperative by adopting a commitment to establish mutual belief with the broker that it also believes the informed proposition. Similar to the earlier reasoning, the leads the distance agent in performing an INFORM to the broker that `bel(distance_agent,done(...))`. This communicative act establishes the mutual belief that the broker is waiting for, thereby, discharging its PWAG that had resulted from the distance agent performing the initial REQUEST. At this point, distance agent is registered with the broker and their commitments and intentions to register the distance agent with the broker have been successfully discharged.

[2]Note that the term 'self' gets replaced by the name of the agent who is sending the INFORM.

(c) Client agent registers with broker and requests it to do the action find_distance

We now start the client agent which registers with the broker in a similar fashion. However, the client agent has two initial high level intentions so it creates two parallel stacks – one for each intention. On the second stack, the client agent ends up with an intention to wait until it is registered with the broker. Once this agent believes that it is registered with the broker, it discharges its intention to wait and comes out of the wait state.

Thereafter, it intends the next action in the intended action expression that happens to be an indefinite repetition consisting of a test action followed by an action `ask_distance`. In each loop of the repetition, the agent intends to do the test action, does the test action (that binds the variables `C1` and`C2` with a city pair), discharges the intention to do the test action, and then intends to do the `ask_distance(C1,C2)` where the variables are bound with the values from the prior test action. Executing this action defined in Table 7.4 results in the client agent having a commitment to have done an action `find_distance(City1,City2,Distance)` where `City1` and `City2` are bound to the same values as `C1` and `C2` in `ask_distance(C1,C2)`. The client agent acts on this PGOAL by trying to execute applicable rules. It first executes the main rule of rational action, which fails to give any actions that can achieve the committed goal. It then executes the second rule of rational action defined in Table 7.2 that results in the client agent having a new PGOAL (relative to the earlier PGOAL) to do an OR action expression whose every branch consists of actions (or action expressions) that can potentially get the broker to do the action `find_distance` that the client agent is committed to getting done. One of the actions in the OR expression is a REQUEST from the client agent to the broker to do the action `find_distance`. Another action in the OR expression is a plan to establish JPG with the broker for doing the action `find_distance`. In both cases, the communication that follows will be the same because the plan consists of a request action followed by a wait by the client agent for the broker to establish mutual belief that it has the interlocking PWAG towards the client agent for establishing the joint commitment. The situation is exactly same as that for registering with the broker except that in that case it was a different action (register action) that the agent was committed to getting done. Similar to that situation, the client agent requests the broker to do the action `find_distance` and the broker agrees to do that action. As earlier, the AGREE communicative act from the broker discharges one of the commitments of the client agent that resulted from its

request to the broker. However, the client agent is still left with a commitment that the broker does the action `find_distance` with respect to client agent's PWAG towards it to do that action. The client agent acts on that commitment by finding a way to be helpful to the broker and finding no way to do so, it waits for the broker to establish mutual belief that the action `find_distance` has been done.

(d) Broker requests distance agent to do action find_distance

The broker acts on its PWAG that followed as a result of it performing the AGREE communicative act for doing the action `find_distance`. That PWAG results in the broker's having a commitment to get the action `find_distance` done. The situation now is exactly similar to that when the client agent had that same commitment. The broker executes the main rule of rational action as a way to act on that commitment. It finds one agent (the distance agent) that is capable of doing that action and intends an OR expression each of whose actions can potentially result in the distance agent's eventually doing the action `find_distance`. We have already discussed how the communication and reasoning progresses in this situation. The broker requests the distance agent to do the action `find_distance(City1,City2,Distance)` where `City1` and `City2` are bound to two specific cities. If the distance agent agrees to do the request, the broker ends up having an intention to wait for mutual belief to be established by the distance agent that the action `find_distance` has just been done.

(e) Distance agent does action find_distance and establishes mutual belief with the broker

The distance agent is now exactly in the same situation as that of the broker after it had sent an AGREE for doing the register action. The PWAG resulting from the AGREE results in the distance agent's having a commitment to do the action `find_distance`. It can do the action find distance itself and so it intends and executes that action. Executing that action binds the third parameter in the action `find_distance(City1,City2,Distance)` with the distance between the two cities from the agent's belief base. Similar to what the broker did after it had done the register action, the distance agent successfully

discharges its intention and commitment for doing the action `find_distance` and re-evaluates its PWAG that had resulted from the AGREE communicative action. It results in the distance agent's having a commitment to establish mutual belief with the broker that the action `find_distance` has been done. The distance agent discharges this commitment similar to how the broker discharged its commitment for establishing mutual belief after it had done the register action. It informs the broker that action `find_distance(City1,City2,Distance)` has been done with the all the three parameters of `find_distance` bound to the appropriate values. The broker confirms by sending an inform that it believes the informed proposition and this discharges the distance agent's PWAG that had resulted from it's agreeing in the first place.

(f) Broker establishes mutual belief with client agent that action find_distance has been done

Recall that the broker was waiting for mutual belief to be established between itself and the distance agent that `find_distance(City1,City2,Distance)` has been done. When the broker confirms the distance agent that it believes the informed proposition that done(`find_distance(...)`), that mutual belief gets asserted into the broker's belief base. As a result, the original stack with intention to do `wait_for(...)` comes out of the wait state and in the process (testing the belief base for that mutual belief) binds the third parameter of `find_distance` with the distance that was found by the distance agent. This also binds the third parameter of `find_distance` wherever it occurs on the stack below the intention to do `wait_for` action. The broker successfully discharges the intention to do `wait_for(...)` along with the commitment to do the action `find_distance` and re-evaluates its PWAG that had resulted from the broker's originally agreeing to the request from the client agent. It results in the broker's having a commitment to establishing mutual belief with the client agent that the action `find_distance(C1,C2,Distance)` has been done. As mentioned above, the Distance parameter is now bound to the distance between the two cities that was found by the distance agent. The situation now is same as that of the client agent's having a commitment to establish the same mutual belief with the broker. The broker intends and executes the plan to establish mutual belief shown in Table 7.5. It sends an inform to the client agent followed by the client agent's confirming that it believes the informed proposition. The resulting mutual belief successfully discharges the broker's PWAG. It also discharges the client agent's intention to wait for

the broker to establish mutual belief that the action `find_distance` has been done and in the process binds the third parameter of `find_distance` on the stack to the value that was found by the distance agent. It also discharges the client agent's commitment for doing the action `find_distance` along with the client agent's intention for doing the action `ask_distance(C1,C2)` that had resulted in this agent committing to do `find_distance`. Thereafter the client agent intends the next loop of the indefinite repetition. The new loop unbinds the variables C1 and C2 in that repeated action expression, the test action returns a new city pair, and the agent intends to do the action `ask_distance` for those two city pairs and the entire process repeats as above.

(g) Case when there is no distance agent and the client agent requests broker to do action find_distance

One scenario is when the client agent is registered with the broker but there is no distance agent in the system. In this case also the client agent requests the broker to do the action `find_distance` as in the above scenario. However, the broker cannot do this action itself and it does not know of any agent who can do this action. Further, the broker does not know of any other broker to whom it can request (as a result of rule second rule of rational action) to do that action. So it will refuse to do the requested action and it will not have any PWAG towards the client agent as a result of the REFUSE communicative act. Furthermore, the REFUSE communicative act will discharge the client agent's PWAG that the broker does the action `find_distance` and has a PWAG towards the client agent with respect to the client agent's PWAG towards it for doing the `find_distance` action. The client agent will retry to achieve its commitment to do the action `find_distance` and will eventually realize that it is impossible to do that action. Similarly, it will infer that it is impossible to do the indefinite repetition (after being unable to achieve it in a few attempts – default value is 3 attempts) and will drop that intention. Similarly, each higher level intention further down the stack will be deemed impossible and will eventually be dropped.

7.1.4 The AAA Fault-tolerance Setup

The AAA fault-tolerance scenario from the previous chapter requires that there be multiple brokers in the system that be reused when one of the brokers becomes unavailable. Next, we briefly examine how the above example of communication between client and distance

agents works when there are multiple brokers in the system.

Two Brokers

In this scenario, we have two broker agents: `broker1` and `broker2`. We make each broker believe that there are two brokers `broker1` and `broker2` by asserting `bel(self,broker(broker1))` and `bel(self,broker(broker2))` in the STAPLE programs for both `broker1` and `broker2`. Further, we make `broker2` register with `broker1` by giving it a high level initial intention to register with `broker1` similar to that of distance and client and agents. We add a rule to the broker programs to make the registration symmetric, that is, when a broker `broker1` registers with another broker `broker2` then both brokers believe that `broker2` has also registered with `broker1`. We modify the distance agent so that it registers with `broker2` and leave the client agent unchanged so that it registers with `broker1` as in the previous scenario.

We first start the distance agent followed by the client agent as earlier. The communication is the same as until the point when `broker1` commits to do the action `find_distance`. In the earlier scenario, `broker1` found an agent (the distance agent) that was capable of doing that action and requested it to do the action `find_distance`. However, in the present scenario, `broker1` does not find any agent capable of doing the action `find_distance` and it cannot do this action itself either. The situation now is exactly the same as that of the client agent when it was working on its commitment to get the action `find_distance` done. The important thing to note is that brokers are just like any other agents and they have access to the same common actions and reasoning rules as any other agents. So `broker1` also uses the second rule of rational action and decides to request the broker that it is registered with `(broker2)` to broker the action `find_distance`. From this point onwards, the communication and reasoning is exactly the same as that which happened when the client agent decided to request `broker1` to do the action `find_distance` except that here `broker1` is in place of the client agent and `broker2` is in the place of broker1. The request from `broker1` goes to `broker2` who agrees to do requested `broker_action.` In turn, `broker2` finds distance agent as a capable of doing the action `find_distance` and requests it to do that action. Distance agent agrees, does that action, and establishes mutual belief that it just did the action `find_distance` as in the above scenario. Now, `broker2` is in the same situation as `broker1` in the previous scenario after the distance agent had established the mutual belief that the action `find_distance` has just been done. It establishes mutual belief with broker1 that the action `find_distance` has just

been done. Thereafter, broker1 establishes mutual belief with the client agent that action `find_distance` has just been done. The variable bindings of `find_distance` are propagated back to the client agent as in the earlier scenario so that the client agent believes what is the distance between those two particular cities. At this stage, both brokers and the distance agent have discharged all their commitments and intentions and the client agent moves on to the next step by intending the next loop of the indefinite repetition as in the previous scenario.

Three brokers

This scenario is similar to the above scenario with two brokers. The setup is same as that in Figure 6.1 where all the three brokers are registered with each other, the client agent is registered with broker2 and the distance agent is registered with `broker3.` This is achieved by modifying the programs of `broker1, broker2,` and `broker3` so that they each believe that they are three brokers – `broker1, broker2,` and `broker3.` We setup `broker2` so that it registers with `broker1` as in the two broker scenario. We also setup `broker3` so that it registers with both `broker1` and `broker2.` The rule for symmetric registration of brokers ensures that all the three brokers are registered with each other.

In this case the behavior is exactly same as that explained in the two broker scenario. When `broker2` does not find an agent capable of doing the action `find_distance,` it decides to request one of the brokers with which it has registered to broker that action (by requesting it do the action `broker_action).` So it may request either `broker1` or `broker3` and the communication and reasoning follows as discussed above.

Alternatively, we could modify the second rule of rational action shown in Table 7.2 so that the OR expression that an agent ends up intending contains a request to all the brokers with which it has registered. If so, `broker2` will choose to request either `broker1` or `broker3` to broker the action `find_distance` as a result of choosing one of the actions in the OR expression and intending it. However, the resulting communication that follows will remain unchanged. We choose to keep the simpler version of the second rule of rational action shown in Table 7.2 for the discussions in the remainder of this chapter.

Next, we discuss what happens when one of the brokers with which an agent is registered in the three broker scenario becomes unavailable.

7.2 IMPLEMENTING AAA FAULT TOLERANCE

We briefly recap the main fault-tolerance behavior of AAA brokers from the last chapter and then explore different ways of implementing the same behavior in the STAPLE brokers discussed above.

7.2.1 Mission statement of AAA Brokers

Table 7.6 shows the first mission statement of the AAA broker team from Section 6.4.1. It says that whenever an agent y registers with the broker team τ, the brokers have a joint intention of reconnecting with that agent, if it ever disconnects, as long as it remains registered with the broker team. An agent is registered with the broker team when is registers with any member of the broker team and it remains registered with the broker team until it explicitly requests a member of the broker team to unregister it. Using the propositions p and q and the action c, the team intention due to that mission statement is given compactly by (JI τ a q) where τ is the agent team and a is the action expression in Table 7.6.

Table 7.6: First Mission Statement of AAA Brokers

$\models \forall$y [(agent y) $\wedge$ (DONE (registered y τ)?) $\wedge$ (dynamic τ)
$\supset$ (JI τ a (registered y τ))] where,

a = (WHILE (registered y τ) DO
[IF $\neg$(connected y τ) THEN (reconnect y τ)])

Let p = (connected y τ), q = (registered y τ), c = (reconnect y τ)
then,a = (q?;[($\neg p$?;c)|p?])*; $\neg q$?

The above mission statement only provided the specification of fault tolerance behavior in AAA based multi-agent systems and the predictions of this specification had to be implemented by hand in Java code for AAA brokers. Recall from Chapter 6 that when an AAA broker disconnects from its teammates (say, when the broker process fails), all the brokers on its team attempt to directly contact the agents that were registered with the now disconnected broker. This behavior is one of the predictions that follow from the above mission statement. Contacting an agent involved connecting with that agent and

requesting it to re-register (i.e., share its capabilities, etc.) with the broker that contacted it. If the agent has already registered (or is in the process of registering) with another broker teammate then it will decline the re-registration request otherwise it will register with this broker. After successfully contacting an agent in this manner, a broker informs this fact to its teammates who then give up their attempts to contact this agent directly. Further, all the remaining brokers in the team are now aware that the agent is registered with the team through that broker. The system configuration is now similar to that in the original situation except that there is one less broker in the system. In contrast, a separate mechanism had to be programmed into AAA agents to recover the communication that was in progress at the time when the failed broker disconnected.

7.2.2 Implementing the AAA Mission Statement in STAPLE

In order to get the brokers written in STAPLE to exhibit the same fault-tolerant behavior as that of the AAA brokers, *all it takes is to specify the mutual belief about the team membership of the broker team τ and the mission statement in Table 7.6.* The facts about team membership can either be provided as part of the broker specification or, in general, the brokers can establish them by dynamically creating the broker team at run time. The dynamic team assumption (dynamic τ) conjunct in the mission statement says that the team will eventually have mutual belief about team membership at all times (Assumption 5.1). It is specified as a separate rule in STAPLE because this assumption is not built into the STAPLE definition of group JPG. The mission statement in Table 7.6 states that the broker team will have the given joint intention whenever an agent registers with the broker team but it does not specify how that joint intention gets established. In AAA, this was implemented (by the AAA programmer) by making sure that the brokers get into a state where they will behave as if they had the above specified joint intention. However, in STAPLE, the brokers will have to establishing any joint intention themselves. Recall that joint intention is nothing but a joint commitment for doing an action mutually believing throughout that the agents are doing the joint action (Definition 2.7 of JI). Also, we have seen that joint commitment can be established using communicative acts, so we will require that the STAPLE brokers establish the joint commitment that corresponds to the joint intention in the AAA mission statement. The action $a$that the broker team jointly intends involves a registered agent and therefore, the joint commitment for (DONE a) must always be established whenever an agent registers with any broker of the broker team. This can be achieved in STAPLE by providing the rule in Table 7.7 to each STAPLE

broker which says that for every agent y had registered with a broker x who is a member of broker team τ, the broker x will have an individual commitment (PGOAL)[3] to establish a joint commitment with the broker team for doing the action of servicing the registered agent as long as the agent y remains registered with the broker team. The action of servicing registered agent as defined by action a is same as that in the AAA mission statement. The action (reconnect y τ) is defined to result in the broker's having an intention to request the agent y to do the action of re-registering with the broker. We also need to replicate the AAA behavior of an agent's refusing to re-register with a broker if it is in the process of re-registering with another broker. This is done in STAPLE simply by specifying an inference rule that says that the agent cannot do the re-register action for a broker if it has already re-registered with another broker or if it is in the process of re-registering with another broker (i.e., if it has a PGOAL, INTEND, or PWAG for re-registering with a broker).

Table 7.7: Mission Statement of AAA Brokers in STAPLE - Version 1

$\models \forall$y [(agent y) $\wedge$ (DONE (registered y x)?) $\wedge$ (member x τ)
$\supset$ (PGOAL x (JPG τ (DONE a(y)) q) q)] where,

p = (connected y τ), q = (registered y τ), c = (reconnect y τ)
a(y) = (q?;**[(¬p?;c)|p?])***; ¬q?

that is,
a(y) = (WHILE (registered y τ) DO
[IF ¬(connected y τ) THEN (reconnect y τ)])

Strictly speaking, the test actions q? and ¬q? in the action expression for a(y) are not needed in STAPLE brokers because q is also the relativizing condition for the joint commitment (JPG) to do a(y) and therefore, the test q? will succeed when the brokers jointly start executing the action expression and the test ¬q? will succeed when the joint commitment is dropped as a result of an agent un-registering with the broker team. So we

[3]The original mission statement in Chapter 6 did not have the PGOAL. In the AAA, the specification of the mission statement was implemented by the agent programmer who made sure that the brokers will end up with a JPG as per the mission statement. However, in this case, the brokers have to establish the JPG themselves. As such, we modified the mission statement slightly to get the brokers to establish the JPG.

simplify the action expression further to that in Table 7.8. The behavior of the resulting broker team will remain exactly the same except that the joint test action $q?$ is no longer executed and therefore, no mutual belief is established that the test action $q?$ succeeded. However, note that because the joint commitment is with respect to q, the team will drop the joint commitment if it ever mutually believes that $\neg q?$ is true. So the indefinite repetition is not really indefinite – it continues only as long as the agent remains registered with the broker team exactly as in Table 7.7.

Table 7.8: Mission Statement of AAA Brokers in STAPLE - Version 2

$\models \forall$y [(agent y) $\wedge$ (DONE (registered y x)?) $\wedge$ (member x τ)
$\supset$ (PGOAL x (JPG τ (DONE a(y)) q) q)] where,
a(y) = REPEAT INDEFINITELY
[IF $\neg$(connected y τ) THEN (reconnect y τ)]

The rule interpreter of the broker teammate with which an agent first registers will fire the rule in Table 7.8 creating a new commitment to establish the given joint commitment. The rule of rational action leads the broker to adopt an intention to establish that joint commitment by executing an action expression (i.e., a plan) consisting of a sequence of group communicative acts a discussed in Chapter 5. Once the joint commitment to do action a is established, each member of the broker team executes the jointly committed action expression as per the discussion in Chapter 6. The communication and coordination, including the starting mutual belief, required to execute the joint action is obtained automatically. It is to be noted that the action expression a does not specify the actors of each constituent action and therefore, the default task allocation algorithm of STAPLE requires all teammates to execute the actions in the action expression a. Also, the STAPLE operational semantics of joint OR expression requires that the team should decide which branch of the OR expression to execute. Again, the required communication follows automatically as discussed earlier in this dissertation (Chapter 4). Once the team decides upon an OR branch, all teammates execute the test action in that branch (which will either be $\neg(connected\ y\ \tau)?$ or $(connected\ y\ \tau)?$) and establish mutual belief about its success or failure in accordance with the operational semantics of joint OR expression. Furthermore, due to the lockstep policy, mutual belief is established when each action in the action expression is done. From Table 7.8, the expression a involves an infinite

repetition and therefore, this mutual belief will be established after each action in every loop indefinitely.

One of the benefits of a programming language such as STAPLE is that many times we can obtain a different behavior just by changing the logical specification. The broker team in this example is interested in finding out when $\neg$(*connected y* τ)? is true, that is, when a registered agent is no longer connected with the broker team. As such, the indefinite repetition of mutually deciding which OR branch to execute in the action expression in Table 7.8 and then executing that action and establishing mutual belief about the findings is really unnecessary and can be construed to be a communication and processing overhead. We can modify this behavior and get rid of the unnecessary communication and processing by replacing the action expression in Table 7.8 by an equivalent action expression shown in Table 7.9 that uses the STAPLE defined no-op action wait_for/1.

Table 7.9: Mission Statement of AAA Brokers in STAPLE - Version 3

$\models \forall$y [(agent y) $\wedge$ (DONE (registered y x)?) $\wedge$ (member x τ)
$\supset$ (PGOAL x (JPG τ (DONE a(y)) q) q)] where,
a(y) = REPEAT INDEFINITELY
[WAIT FOR $\neg$(connected y τ) THEN (reconnect y τ)]

The execution of STAPLE defined action `wait_for(`p`)` takes an indefinite amount of time – this action completes when p, i.e., (*connected y* τ) becomes true. The agent can execute other actions and act on its other commitments and intentions while it is waiting for (*connected y* τ) to become true meaning that the agent itself is not in a wait state. In other words, only the stack which has the intention to execute `wait_for(p)` goes into a wait state until (*connected y* τ) becomes true. The agent sets a trigger on its belief base for (*connected y* τ) to become true and that stack comes out of the wait state whenever this trigger fires. The action expressions in Table 7.8 and Table 7.9 are equivalent in the following sense. In the first case, the brokers will keep testing (*connected y* τ) and finding that (*connected y* τ) is true indefinitely until the agent disconnects (in which case (*connected y* τ) becomes false). They will reconnect with the disconnected agent and then again indefinitely keep testing and finding that (*connected y* τ) is true, and so on. However, in the second case, the brokers know that (*connected y* τ) is true to start with and they don't keep testing for it indefinitely, rather, they wait until they detect that

(*connected y* τ) is false, that is, until the agent disconnects. They will then reconnect with the disconnected agent thereby making (*connected y* τ) true and then they will again wait for p to be become false instead of continuously testing for it as team. In fact, the test whether (*connected y* τ) is true or false happens in both cases. In the first case, the broker team consciously decides to do the test, tests for it, and establishes mutual belief about the result. In the second case, each agent's belief base monitors p(*connected y* τ) behind the scenes and there is no team decision making and communication about testing (*connected y* τ) until at least one broker teammate detects that (*connected y* τ) is false.

We can further simply the action expression in Table 7.9 by specifying the actors of the two actions `wait_for(p)` and `reconnect`. Table 7.10 shows the translation of this specification in STAPLE syntax where we have specified that every broker who intends the action expression `service_registered_agent` does the actions `wait_for` and `reconnect`. If we replace "self" by one of the brokers (say, broker1) in the `wait_for` action or `reconnect` action then that broker (say, broker1) will end up doing that action as part of the joint action `service_registered_agent`. Similarly, if we replace `'self'` with `'any'` or `'all'` or `'brokerteam'` then each of these will result in a potentially different behavior depending on the team decision making process. The predicate `assert_once` in this table prevents duplicates when asserting a proposition into the agent's belief base.

Table 7.10 first specifies the mutual beliefs about the broker team – it says, that it is mutually believed that there are three brokers in the broker team viz `broker1,` `broker2,` and `broker3`. Strictly speaking, it is not necessary to specify these mutual beliefs. However, in that case, we would have to add a rule to deduce that there is a new member in the broker team whenever a broker registers with another broker (and to update the belief base accordingly). This deduction rule in conjunction with the dynamic team assumption (that establishes mutual belief in the team whenever a new team member joins the team or when an existing team member leaves the team) will establish these mutual beliefs about broker team membership at run time. In this example, we will specify these mutual beliefs in each broker program instead of the deduction rule (so any new brokers other than those already specified cannot be added to the system).

Thereafter, Table 7.10 specifies a named action expression (a plan) for the AAA mission statement as discussed above. The precondition of the plan `service_registered_agent` illustrates another flexibility of STAPLE programs. Here the precondition is used for making sure that a broker who is committing to service an agent registered with a broker

Table 7.10: STAPLE Encoding of AAA Broker Fault-Tolerance - Part 1

```
% Beliefs about broker team
bel(self,team_name(brokerteam)).
bel(self,mb(brokerteam,team_member(broker1,brokerteam))).
bel(self,mb(brokerteam,team_member(broker2,brokerteam))).
bel(self,mb(brokerteam,team_member(broker3,brokerteam))).

% The action expression for AAA Mission statement in Table 7.9

plan(service_registered_agent,6) :-
    [description:  'Plan to service a registered agent',
    args:  [Broker,Agent,Host,Port,E1,E2],
    precondition:  {(bel(self,team_member(Broker,brokerteam)),
                  assert_once(registered(Agent,Broker)),
                  assert_once(servicing_agent(Agent)))},
    body:  {(action(wait_for(~connected(Agent)),self,E1),
             action(reconnect(Agent,Host,Port),self,E2))..
             },
    effects:  [(serviced_registered_agent(Agent,Host,Port),1.0)]
].

% Mission statement of AAA brokers encoded as a rule to establish JPG
% to service registered agent when an agent registers with a broker

rule(rule1,done(action(register(agent(Agent,Host,Port)),self)),5) :-
  \+ bel(self,team_member(Agent,brokerteam)),
  \+ bel(self,servicing_agent(Agent)),
  assert(servicing_agent(Agent)),
  Action = establish_jpg(brokerteam,done(action(service_registered_agent(
self,Agent,Host,Port,A1,A2),brokerteam)),registered(Agent,brokerteam))
  ==>
  adopt(intend(self, Action, true, 10.0)).
```

teammate believes that the agent is registered with that teammate. A better place for asserting the fact `registered(Agent,Broker)` is the belief base maintenance system where we can define a rule to deduce that an agent is registered with a broker teammate who has established a mutual belief with the broker team that it has a PWAG towards the broker team that the team adopt a PWAG to service a registered agent. Another place to add this assert statement is during the processing of a received message by adding a rule to deduce that the agent is registered with a broker teammate who is attempting to establish a JPG with the entire team to service a registered agent (say, via a REQUEST communicative act). However, we assert this fact in the precondition of the plan `service_registered_agent` just to demonstrate this flexibility.

The rule in Table 7.10 is the rule in AAA mission statement. It says that if a broker has done the action of registering an agent and if it does not believe that the agent is being serviced by the broker team then it adopts an intention to establish a JPG with the broker team to service that registered agent. Again, this rule can be written in several different ways, for instance, by checking that there is no JPG in the team yet for servicing the registered agent (and there is no intention to establish that JPG either) instead of checking whether or not it believes that the agent is being serviced. The reconnect action used by the brokers to reconnect with a disconnected agent who is already registered with the broker team is defined in Table 7.11. This action simply results in the broker having an intention to request the disconnected agent to do the action `register_with_broker` specifying itself as the broker with whom the agent should (re)register. Table 7.11 also shows a belief base maintenance system rule `(reduce_assert/1)` to deduce that agents that were registered with a disconnected broker teammate are no longer connected and to update the belief base accordingly.

The STAPLE code in Table 7.10 and Table 7.11 is added to the STAPLE programs for each broker (i.e., `broker1, broker2,` and `broker3`). Just with these specification level changes, the STAPLE brokers will now exhibit a fault-tolerance behavior similar to that of the AAA broker team as discussed next.

7.2.3 The AAA Fault-tolerance Example in STAPLE

We first set up three brokers similar to that in Figure 6.1 as discussed in Section 7.1.4. Then we setup the client agent so that it registers with broker3 and start it. The communication and reasoning for registering happens just as described in Section 7.1.3. However, as soon as the broker does the register action, the reactive rule for mission statement of

Table 7.11: STAPLE Encoding of AAA Broker Fault Tolerance - Part 2

```
% The reconnect action used in the plan service_registered_agent
action_definition(reconnect,3) :-
    [args:  [Agent,Host,Port],
    code:  { assert_once(agent_address(Agent,Host,Port)),
             Action=request(self,Agent,
                    action(register_with_broker(self),Agent,Eid),true),
             subgoal(intend(Action),StackId)
           },
    stack_id:  StackId,
    effects:  [(<>reconnected(Agent),1.0)]
    ].
% Belief base maintenance system rule to deduce that agents registered
% with a disconnected broker teammate are not connected
reduce_assert(self,~connected(Agent)) :-
    assert_once(~connected(Agent)),
    bel(self,team_member(Agent,brokerteam)),
    reduce_assert(~team_member(Agent,brokerteam)), %fires dynamic
                                                   %team assumption rule
    findall(X,(retract(registered(X,Agent))),ListOfDisconnectedAgents),
    for(Y,ListOfDisconnectedAgents,assert(~connected(Y))).
```

broker team (in Table 7.10) fires resulting in `broker3` intending the following new high level intention to establish JPG in the broker team to service the registered agent (the distance agent) as long as this agent remains registered with the broker team.

intend(broker3,action(establish_jpg(brokerteam, done(action(service_ registered_agent (broker3,distance_agent,localhost,5045,...), registered(distance_agent, brokerteam)),...)

STAPLE agents have access to an action expression (i.e., to a plan) to establish JPG in a team. This plan is an implementation of the interlocking PWAG theorem – its body consists of a REQUEST from the agent to a team followed by the agent waiting for mutual belief to be established in the team that the team has a PWAG towards the agent with respect to the agent's PWAG towards the team. As such, `broker3` intends to request the broker team to do the action service_registered_agent relative to the proposition that the distance agent is registered with the broker team. Thereafter, it performs this request communicative act resulting in the following message being sent to all members of the broker team.

request(broker3,brokerteam,action(service_registered_agent(broker3, distance_agent, localhost,5045,...),registered(distance_agent,brokerteam)).

Note that this request from `broker3` is addressed to the `brokerteam` rather than to

any member of the broker team. The communication actuator of this broker evaluates the team membership by checking its belief base and finds that there are three members in the team – `broker1, broker2,` and `broker3.` As such, `broker3` sends this request message to all the team members other than itself. Both `broker1` and `broker2` receive exactly the same physical request message shown above that corresponds to the request communicative act. Both `broker1` and `broker2` decide to agree to the group request. Thereafter, `broker2` sends the following group AGREE that is again addressed to the `brokerteam` rather than to any specific member of the team.

gagree(broker2,brokerteam,broker3,action(service_registered_agent(. . .),. . .)

Also, the physical `gagree` message gets sent to all team members, other than `broker2` itself, rather than to just the original requester `(broker3)`. Similarly, broker1 sends a group agree to the broker team. From Lemma 5.5, this is sufficient to establish JPG in the broker team for doing the action `service_registered_agent`. This JPG discharges `broker3`'s intention to wait for the team to have a PWAG towards it for doing the action `service_registered_agent` because this PWAG follows from the group JPG. Each of the three brokers adopts the PWAG that follows from the group JPG as a new high level social commitment towards the `brokerteam`. For instance, `broker2` has the following PWAG on a new stack:

pwag(broker2,brokerteam,done(action(service_registered_agent(. . .)),jpg(brokerteam,done (action(service_registered_agent(. . .))))

As in other STAPLE examples, the PWAG leads to individual commitment for doing the jointly committed action. The action `service_registered_agent` is defined as an action sequence in Table 7.10 consisting of waiting for the agent being serviced to disconnect followed by reconnecting the disconnected agent. As such, each broker ends up having an intention to wait for the distance agent to disconnect. Figure 7.1 shows the stack of one of the brokers as a result of the group JPG. Recall that the relativizing condition for an item on the stack is a conjunction of any explicit relativizing condition shown in that term with the term on the stack immediately below it. For example, the arrows in Figure 7.1 indicate that the explicit relativizing condition "true" means that the relativizing condition is the item on stack item pointed to by the arrows.

The commitment stack of `broker1` (and `broker3)` for servicing the distance agent is similar to that in Figure 7.1 except that the commitments on the stack are for `broker1` (and `broker3)` instead of `broker2.` Similarly, when we start the client agent, it registers with `broker1`, which then establishes JPG in the broker team for servicing the client

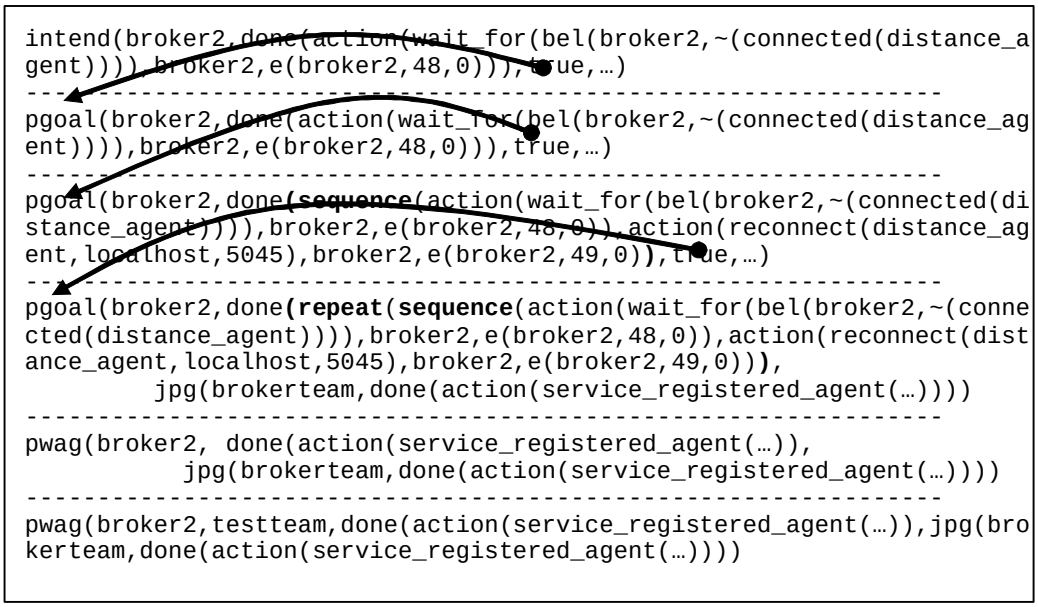


Figure 7.1: Commitment stack of broker2 as a result of AAA mission statement

agent. Each broker ends up with a stack for servicing the client agent similar to that for the distance agent. This is the steady state in the AAA fault tolerance scenario. The client agent repeatedly sends request for the action `find_distance` to `broker1` which results in a request to the distance agent from `broker3` and the answer from the distance agent is routed back to the client agent (Section 7.1.3). The request from the client agent and subsequent answer back to it results in separate set of stacks as commitments and intentions get created and discharged as discussed earlier.

At this point, we terminate `broker3` by killing the operating system process that corresponds to this broker. Distance agent is now disconnected from the broker team and there is no way for requests from the client agent to be routed (i.e., to be brokered) to the distance agent. However, the system recovers from this failure in a similar manner to the AAA brokers in virtue of the encoding of AAA mission statement as discussed so far.

Specifically, `broker2` notices that `broker3` has been disconnected because `broker2`'s network observer detects the lost network connection with broker3 and adds `bel(self,` $\sim$`connected(broker3))` to its belief base. The belief base maintenance system of `broker2` retracts `bel(self,group_member(broker3,brokerteam))` from its belief base. The rule for dynamic team assumption fires resulting in `broker2` adopting a new high level intention to establish mutual belief in the team that `broker3` is no longer a member of the broker team.

Further, using the belief base maintenance system rule `(reduce_assert)` in Table 7.11 broker2 infers that all agents that were initially registered with `broker3` are now disconnected and therefore, it asserts `bel(self,`$\sim$`connected(distance_agent))` into its belief base. This new belief fires a trigger set by the intention term on top of the stack in Figure 7.1 and that stack comes out of wait state. The intention and commitment to wait for $\sim$`connected(distance_agent)` in Figure 7.1 are successfully discharged and the broker re-evaluates the commitment for doing the action sequence in that figure. However, before it can commit to the next action in the action sequence, it must establish a mutual belief in the team (due to the `lockstep` policy adopted by the broker team) that the first action in the joint action expression has just been done. As such, broker2 now adopts a commitment to establish a mutual belief in the broker team that the action wait for $\sim$`connected(distance_agent) has been done.`

Now, broker1 also detects that `broker3` is disconnected and it reasons in a similar manner as `broker2.` Therefore, at this point, both brokers `(broker1` and `broker2)` have individual commitments to establish two different mutual beliefs in the broker team. One commitment is to establish mutual belief that `broker3` is no longer a member of the broker team. The other commitment is to establish mutual belief that the action wait for $\sim$`connected(distance_agent)` has been done. These mutual beliefs are established in the usual manner as discussed earlier in this dissertation, say, by intending the plan to establish mutual belief in Table 7.5. An interesting observation is that since both agents are trying to establish mutual belief about the same fact at the same time, they inform each other that p and there is no separate confirmation message that the informed proposition is believed by the recipient. This is because (INFORM x y p) and (INFORM y x p) establishes the same mutual belief as (INFORM x y p) followed by (INFORM y x (BEL y p)).

As soon as the mutual belief that the "wait for" action has been done, the joint execution of the action expression in Figure 7.1 moves to the next action in the action sequence. At this point, both broker1 and broker2 have a commitment for doing the action reconnect(distance_agent, localhost,5045). From Table 7.10 and Figure 7.1, we see that each broker (self) is the actor for the reconnect action. So each broker intends the reconnect action and executes the definition of this action given in Table 7.11. From Table 7.11, when `broker2` executes the reconnect action, it ends up with an intention to request the distance agent to (re)register with itself (as part of reconnecting with the broker team). It performs this request communicative act that is then analyzed by the

distance agent. In the meanwhile, broker1 is doing exactly the same thing and so it also requests the distance agent to register with itself. The exact communication and reasoning that follows next depends on timings of the communication.

Let us assume that the distance agent gets the request from `broker2` before it gets the request from `broker1`. If so, it agrees to `broker2's` request and it refuses `broker1's` request (because the precondition of the action `register_with_broker` in Table 7.2 fails when it analyzes broker1's request). This precondition checks that the agent does not already have an intention to register with a broker. Thereafter, the distance agent acts on the requested action that is has agreed to do and (re)registers with the `broker2`. Due to the lockstep policy, `broker2` intends to establish mutual belief that the action of reconnecting with the distance agent has been done. As such, it informs `broker1` that `done(reconnect(distance_agent,...))` to which broker1 responds that it believes the informed proposition thereby, establishing the mutual belief required by the lockstep policy. The execution of the joint action expression moves to the next step and both broker1 and broker2 intend to wait for $\sim$`connected(distance_agent)` as before. At this point, the distance agent has been reconnected with the broker team and the requests of the client agent can be routed to it once again. Next, we discuss what happens to the requests that were in progress when a broker becomes unavailable.

Recovering ongoing requests

The recovery of an ongoing request is achieved by implementing a rule to reconsider commitments with a disconnected agent. This rule fires whenever an agent believes that another agent that was earlier connected with it is no longer connected. If the distance agent was in the middle of responding to `find_distance` when broker3 got disconnected, the distance agent will reconsider its PWAG towards `broker3` for doing the action find_distance and drop that PWAG believing that it is impossible to establish any mutual belief with `broker3`. Suppose that the request to broker the action `find_distance` had come to `broker3` from `broker1`. In this case, `broker1` will similarly reconsider its PWAG towards `broker3` for doing the action `broker_action` and drop that PWAG believing that it is impossible to establish any mutual belief with `broker3`. Thereafter, it will retry to achieve its commitment for doing the action `find_distance` and will ask `broker2` to "broker" that action. Eventually, `broker2` will establish mutual belief with `broker1` that it is impossible to do broker the action `find_distance` because it cannot do that action `find_distance` itself and it does not know of any agent capable of doing that action.

Thereafter, `broker1` will establish mutual belief with client agent that it is impossible to broker the action `find_distance.` Basically, the interaction is the same as if the `broker1` did not know of any capable agent or any other broker when the client agent was started.

On the other hand, if we had terminated the broker with which the client agent was registered `(broker1)` then the request in progress when the broker1 became unavailable can be recovered by the client agent simply by adding a rule to wait for connection with a broker if the agent is currently not connected with a broker. In this case, the client agent will drop its intention to request `broker1` for brokering the action `find_distance` and it will reconsider its commitment that led to this intention. The new rule to wait for connection with a broker will fire when the client agent is reconsidering its PGOAL for the action `find_distance` and that stack will end up waiting for the client agent to be (re)connected with a broker. Eventually, when the client agent reconnects with a broker, this stack will come out of the wait state and the client agent will request the newly connected broker to "broker" the action `find_distance.`

Next, we discuss a benefit of direct execution of logical specifications in STAPLE wherein agents written in STAPLE were able to correctly handle a situation that was unplanned for by the agent programmer.

Handling unplanned situations in STAPLE

There is one situation that we had not planned for during the design and implementation of the AAA brokers. As such, the expected behavior of the AAA brokers in that situation had not been predicated from the fault-tolerance specification and therefore, had not been implemented into the AAA brokers. However, the STAPLE brokers logically reasoned through that situation and handled it correctly.

This situation involved subtle timing of synchronization messages (to establish mutual beliefs) in the broker team when a broker teammate discovers that a disconnected agent has successfully re-registered with some broker teammate. In the AAA implementation, the broker teammate with whom a disconnected agent successfully re-registered establishes the mutual belief about that fact, thereby discharging the team commitment to reconnect to the disconnected broker. All other teammates assume that the broker who established that mutual belief is the one with whom the agent is reconnected. In almost all cases, this is what would actually happen because the broker teammate with whom the disconnected agent re-registered is the first one to learn that the disconnected agent is now reconnected. However, it is possible (though not very frequent) that another broker teammate who

learns of the fact that the agent is reconnected with a broker teammate ends up establishing mutual belief about this fact either before or at nearly the same time as the broker with whom the agent actually reconnected.

As in the above example, we assume that the distance agent gets the request for doing the action `register_with_broker` from `broker2` before it gets the same request from `broker1`. However, this time assume that the delay between the two requests is much longer and that the distance agent is already registered with `broker2` when it gets the request from `broker1`. As such, the effect of the requested action is already true and therefore, the distance agent informs `broker1` that it is already registered with a broker. This creates a situation in which both `broker1` and `broker2` try to establish mutual belief in the broker team that the distance agent is registered with the broker team. It leads to conflicting mutual beliefs because the brokers infer that an agent is connected with the broker who establishes the mutual belief about that agent's registration with the broker team. The AAA brokers were not programmed to handle this situation, but the STAPLE brokers correctly dealt with the situation – the conflicting mutual beliefs got resolved when the broker with whom the agent reconnected (`broker2` in this example) established another mutual belief that *it* had done the action of reconnecting with distance agent. It shows that direct execution frameworks like STAPLE can deal with a certain category of surprise wherein a designer or programmer may not have foreseen a situation while translating the specification into design and eventual implementation.

Next, we modify the mission statement of STAPLE brokers slightly to get a different behavior.

7.2.4 Modifying the AAA fault-tolerance behavior

One problem with the above AAA type behavior is that the attempt by both `broker1` and `broker2` to reconnect with the distance agent at the same time is not optimal. Further, it can run into a different problem if broker1 fails to reconnect with the distance agent several times and infers that it is impossible to reconnect with the distance agent. If so, it will try to establish mutual belief in the broker team that it is impossible to reconnect with the distance agent while `broker2` will try to establish mutual belief in the broker team that it has successfully reconnected with the distance agent. The STAPLE brokers would need to be provided with inference rules to deal with this situation. As we have seen before, the behavior of STAPLE agents can be changed by simply changing the joint action expression.

We modify the action expression of the AAA mission statement encoded in Table 7.10 by changing the actor of the reconnect action from "self" to "any". Now, the second action in the joint action expression is to be done by any one member of the team and therefore, it requires a team decision on who will do the reconnect action. Various team decision making algorithms can be plugged in to enable the team to decide the actor of the reconnect action. The default STAPLE algorithm is similar to that for the joint OR action expression – a team leader (decided by convention from among the team members) picks one of the team members as the actor and establishes mutual belief about that fact. This team member then reconnects with the distance agent and establishes mutual belief about when it is done. This behavior alleviates both of the problems with the original behavior of AAA brokers as pointed above.

7.3 Conclusion

We implemented the fault-tolerance of AAA brokers in STAPLE agents and we reproduced the behavior of AAA brokers simply by providing the high level specification of AAA type fault-tolerance to STAPLE brokers. We showed that the automatic reasoning and communication that occurs during steady state as well as during the recovery process is as predicted by the JI theory. We also demonstrated how modifying the logical specification of behavior at a high level results in very different agent and team behavior. We saw one instance wherein a STAPLE based system were able to handle a situation that was unaccounted for in AAA. To provide a rough statistics, the implementation of brokering and fault-tolerance (due to Mission Statement 1) in AAA took more than 2500 lines of Java code whereas the same behavior took less than 50 lines of STAPLE code (less than 20 logical sentences).

Chapter 8

Conversations for Teamwork

Teamwork and communication go hand in hand. Communication is required not only for establishing and discharging teams but also for coordination and synchronization purposes. Multi-agent communication seldom consists of a single communicative act, rather agents engage in extended conversations consisting of several communicative acts in order to achieve their communicative goals. For example, when an agent requests another agent to do an action, the other agent either accepts or refuses, and upon accepting, it eventually informs the first agent about the success or failure of the requested action. Conversation protocols specify the sequence of allowed communicative acts and are critical to the interoperation of agents with different reasoning abilities. Some agents may have the ability to reason using the formal semantics of communicative acts, and others may be programmed to respond to communicative acts in an event-response fashion. However, these two kinds of agents can still communicate with each other in a meaningful way if they follow well-defined conversation protocols. For instance, in the above example, the second agent knows that it has to respond to a REQUEST with either an AGREE or a REFUSE and the first agent knows that it will receive either an AGREE or a REFUSE in response to its request. Therefore, even if the two agents have different capabilities with regard to reasoning about communicative acts, they can still communicate effectively with each other as long as they follow this simple protocol and know how to respond to (and with) these communicative acts. In this chapter, we lay out a framework for specification and analysis of conversation protocols that will allow agent designers to specify correct protocols on one hand, and on the other, will provide a means for STAPLE agents to reason about conversation protocols.

8.1 OVERVIEW

Conversation protocols represent communication patterns in multi-agent interactions. These protocols are traditionally specified as finite state machines in which the transition arcs specify the communicative actions to be used by the various agents involved in a conversation. Figure 8.1 shows a fragment of the request protocol mentioned above. In this figure, 'Start' and 'End' represent the start state of the protocol and its possible final states respectively, and S1 and S2 represent some intermediate states. The transition arcs marked with REQUEST, AGREE, REFUSE, and INFORM specify the communicative actions used in this protocol.

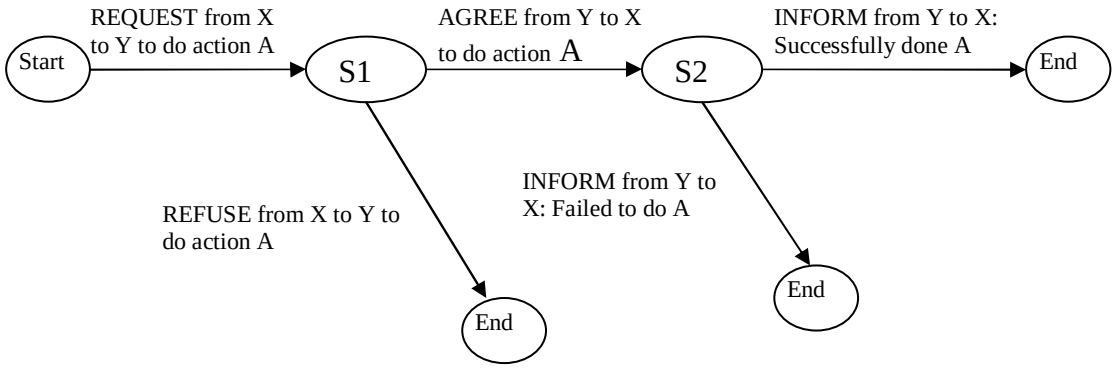


Figure 8.1: A Sample Conversation Protocol

Protocols are executed by performing the communicative actions and therefore, the communicative actions have come to be regarded as the central concept around which analyses of protocols are based. However, it is the states and not the state transitions that are key to the correctness and completeness of a protocol [106, 123]. This dissertation introduces a landmark-based approach for formal analysis of conversation protocols wherein the most important aspect of a conversation protocol is not the communicative actions involved in that protocol but the effects or the states that these actions bring about. The basic idea is that, since protocols are used to achieve certain tasks or to bring about a certain state of affairs in the world, one might identify the important landmarks or states of affairs or subtasks that are brought about by and during the execution of a protocol. Conversation protocols can then be expressed at an abstract level as partially ordered landmarks where each landmark is characterized by propositions that are true in the state represented by that landmark. Several different actions can bring about the

same state and therefore, the partially ordered landmarks represent a family of protocols. Communicative actions are the tools to realize concrete protocols from a landmark-based representation. Besides contributing to formal analyses of protocol families, the landmark-based representation facilitates techniques similar to partial order planning [73] for dynamically choosing the most appropriate action to use next in a conversation, allows compact handling of protocol exceptions, and in some cases, even allows short-cutting a protocol execution by opportunistically skipping some intermediate landmarks.

Limitations of the finite state machine representation for protocols have led agent researchers to explore various other techniques such as Definite Clause Grammars [68], Colored Petri-Nets [35], and enhanced Dooley graphs [82]. Each of these alternatives attempts to remedy some shortcoming of the finite state machine representation for the purpose of representing and reasoning about conversation protocols. For instance, the enhanced Dooley graphs of [82] are a directed graph similar to finite state machines with a few significant differences: states are annotated with a "character", or role, that an agent plays within a conversation; transitions are represented such that the evolution of a conversation is modeled explicitly rather than simply indicating possible transitions at some point during a conversation. However, these techniques have not been widely adopted by the multi-agent community. Agent researchers need a formalism for protocols suitable for automated reasoning and an easily understood visual representation of conversation protocols. Recent efforts have been made to extend the Unified Modeling Language (UML) for a visual representation of protocols that can be used by software developers. The resulting specification called AUML (Agent UML) [5] is promising and will mature over time. But the need for a proper formalism for conversation protocols still remains. A formalism for concrete protocols is suggested by the very definition of conversation protocols as a pattern of communicative actions. We represent concrete protocols along with their associated precondition and goal as action expressions using dynamic logic constructs. These communicative action expressions involve multiple cooperating agents and are joint action expressions. The motivation for joint action expressions also comes by analogy with natural language wherein dialogues are treated as joint actions [45, 46, 27, 31, 21]. Recently, other researchers have also started looking at conversation protocols as a joint activity [40, 113] by analogy with natural dialogue. The communicative actions in the joint action expression for a protocol achieve the landmarks of that protocol in the required order. This chapter explores the application of joint intention theory [70, 28] to protocols represented as joint action expressions and presents a way to analyze conversation protocols

and their compositions.

We present the landmark-based approach for representing and analyzing families of protocols in Section 8.2, and apply this technique to three different protocols in Section 8.3. In Section 8.4, we apply the joint intention theory to protocols, and in Section 8.5, we introduce the joint action expression based representation for protocols and present a formal analysis of protocol compositions. Finally, we discuss execution of conversation protocols in STAPLE in Section 8.6 and conclude in Section 8.7 with a summary of this chapter.

8.2 PROTOCOLS AS PARTIALLY ORDERED LANDMARKS

Conversation protocols are traditionally specified as finite state machines in which the transition arcs are labeled by the communicative actions that cause those transitions. One of the problems of this specification is that the communicative actions to be used in a conversation are fixed by the protocol. However, there may be several (communicative as well as non-communicative) actions that result in the same transition. Also, a specified communicative action in a protocol may not be applicable in every situation due to factors such as different prerequisites, side effects, costs, and time to completion. We want a way of specifying conversation protocols that allows an intelligent agent to choose the best applicable action dynamically in any situation. We propose a landmark-based approach for formal analysis of conversation protocols wherein the most important aspect of a conversation protocol is not the set of communicative actions involved in that protocol but the effects or the states that these actions bring about. The proposed representation looks similar to state machines but instead of specifying the state transitions, it specifies a partially ordered set of states. The basic idea is that since protocols are used to perform certain tasks or to bring about certain states in the world, one might identify the important landmarks or states that are brought about by and during the execution of a protocol. Conversation protocols can, then, be expressed at an abstract level as a set of partially ordered landmarks, where each landmark is characterized by a conjunction of propositions that are true in the state represented by that landmark. Several different actions can bring about the same state and therefore, the partially ordered landmarks represent a family of protocols. Communicative actions are, then, the tools to realize concrete protocols from a landmark-based representation.

We do not specify transitions in landmark-based analysis and so an agent may use any means to transition from one landmark to the next partially ordered landmark. It will typically perform some actions and these actions may or may not be communicative actions. In order to help the initiator, an agent may be able to opportunistically skip various landmarks during protocol execution and try to reach a later landmark by performing some action that can achieve it. A participating agent may also try to skip intermediate landmarks during protocol execution to help the initiator if it knows the global intentions from the goal of the protocol being used. The short-cutting of a protocol execution requires that landmarks be labeled as either required or optional, and an agent may skip only the optional landmarks. A landmark-based representation can also be specialized by inserting additional landmarks, which further constrain the set of possible transitions. We now illustrate these concepts in further detail.

8.2.1 Visual and Logical Representation

One can visualize a landmark-based representation of protocol families as a directed graph whose nodes represent landmarks and whose directed edges represent partial ordering. This representation looks similar to finite state machines except that the transition arcs specify only the ordering of the states and not the communicative acts required for the state transitions. Figure 8.2 shows a protocol family using partially ordered landmarks. L1 is the initial landmark (the start state) represented by two concentric hexagons, L5 and L6 are final landmarks represented by dark hexagons, L2 and L4 are important intermediate landmarks represented by solid hexagons, and L3 represented by a dotted hexagon is specified as an optional intermediate landmark. The arrows indicate ordering of the landmarks – L1 comes before L2 and L4, L2 comes before L3 and L3 comes before L5 and L6 and so on. We call this ordering partial because there are some landmarks (such as L2 and L4) that do not have a hard ordering relationship between them, and furthermore, one may insert additional ordered landmarks between any two landmarks. The landmark-based representation specifies the waypoints from the initial landmark to one of the final landmarks. Optional landmarks in any path may be skipped opportunistically *during protocol execution* but the important landmarks must be followed. Following a path means performing actions from one landmark to reach either the next landmark or the one after, skipping any number of optional landmarks. The allowed actions to transition from one landmark to the next may be either a single action (communicative as well as non communicative action) or a complex action expression consisting of several actions.

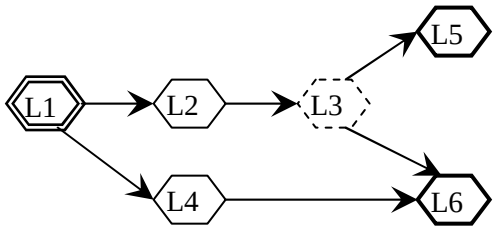


Figure 8.2: Partially Ordered Landmarks

A protocol family may or may not specify a landmark as optional depending on the goal of the protocol. For instance, consider a request protocol that has an intermediate landmark in which the requester and the requestee are jointly committed to doing the requested task. Suppose that I request you to open the door using this request protocol. Let us assume that you recognize the goal and intention of my request and decide to be helpful. If the intermediate landmark requiring joint commitment is optional then you may go ahead and open the door without further communication. However, if that intermediate landmark was not specified as optional then you must first communicate with me to establish joint commitment between us before you open the door.[1] Also, note that making a landmark of a protocol family optional may make all protocols based on that family fragile if the optional landmark requires establishing joint commitment as in this example. This is because there is no joint commitment between the agents, to take care of exceptional situations, if an agent opportunistically skips that optional landmark during protocol execution. So making landmarks optional has both advantages (speed) and disadvantages (fragility). Therefore, if an agent prefers speed to resilience for some task, then it may use a protocol with such an optional landmark for that task. But if fault-tolerance is of utmost importance, then one should use a protocol based on a protocol family in which the landmarks requiring team formation are not optional.

Protocol families expressed using landmarks can be represented logically in a dynamic logic that can express the temporal ordering of propositions. Here, we introduce a simple propositional dynamic logic sufficient for our purpose and use it to logically represent the protocol family in Figure 8.2. We use temporal operators PRIOR, HAPPENS, and eventually ($\Diamond$), and operators for action sequences ($a;b$) and test action ($p?$), from the logical language of joint intentions (Chapter 2), to define the operators for landmark ordering.

[1] In either case, if the final landmark requires mutual belief that the requested task has been done, then you are required to establish that mutual belief for protocol termination.

Let $\mathbb{L}$ be the domain of landmarks, let $\mathbb{P}$ be the set of atomic propositions, and let $\mathcal{P}$ a function that gives the conjunction of atomic propositions comprising a landmark, that is,

$\mathcal{P} : \mathbb{L} \rightarrowtail \mathbb{P}$

Let L1, L2, and L3 be three landmarks. We define a partial order operator $\succ$ such that L1$\succ$ L2 means that L1 comes prior to L2 but not necessarily immediately before L2. Also, two landmarks can be ordered using the partial order operator only if they are different landmarks. The partial order operator is defined inductively as follows.

Definition 8.1. *Partial Order Operator*

$$\text{L1} \succ \text{L2} \triangleq (\text{PRIOR } \mathcal{P}(\text{L1})\ \mathcal{P}(\text{L2}))\ \wedge (\mathcal{P}(\text{L1}) \neq\ \mathcal{P}(\text{L2}))$$
$$\text{L1} \succ \text{L2} \succ \text{L3} \triangleq (\text{L1} \succ \text{L2}) \wedge (\text{L2} \succ \text{L3})$$

where, from [26]

$$(\text{PRIOR}^2 p\ q) \triangleq\ \forall c\ (\text{HAPPENS } c; q?) \supset\ \exists a\ (a \leqslant c) \wedge (\text{HAPPENS } a; p?)$$

That is, a proposition p occurs prior to proposition q if for all event sequences c such that c occurs after which q is true, there exists an *initial subsequence* a of c such that a occurs after which p is true. This definition does not say whether p or q will ever be true. However, if q is ever true then there must be some earlier time when p was true. Formally,

$$\models\ (\text{PRIOR } p\ q) \wedge\ \Diamond q \supset\ \exists e\ (\text{HAPPENS } p?; e; q?)$$

Combining this proposition with Definition 8.1, we get the following property about landmark ordering.

Proposition 8.1.

$$\models (\text{L1} \succ \text{L2}) \wedge\ \Diamond \mathcal{P}(L2) \supset\ \exists e\ (\text{HAPPENS } \mathcal{P}(\text{L1})?; e; \mathcal{P}(\text{L2})?) \wedge (e \neq nil)$$

That is, if landmark L1 comes before landmark L2 and if propositions in landmark L2 will eventually be true, then there exists a non-empty event sequence e of primitive event types such that e occurs after propositions in L1 are true and after e occurs, the propositions in L2 will be true. The empty event sequence *nil* is a subsequence of all event sequences.

[2]PRIOR was called BEFORE in [25]. However, we use the term PRIOR here to avoid confusion with a different usage of the term BEFORE in this paper.

From Definition 8.1, it follows that the partial order relation is transitive, that is, if landmark L1 comes before landmark L2 and landmark L2 comes before landmark L3 then landmark L1 comes before landmark L3.

Proposition 8.2. $\models (\text{L1} \succ \text{L2} \succ \text{L3}) \supset (\text{L1} \succ \text{L3})$

An OR operator $\perp$ over landmarks is similarly defined in terms of propositions that are true in those landmarks.

Definition 8.2. *OR Operator*

$\text{L1} \perp \text{L2} \triangleq \mathcal{P}(\text{L1}) \vee \mathcal{P}(\text{L2})$

This definition says that the current landmark is either L1 or L2 means that the propositions in landmark L1 or the propositions in landmark L2 are true. The following properties are a consequence of Definition 8.1, Definition 8.2, and Proposition 8.1 above and the definition of HAPPENS (Chapter 2).

Proposition 8.3.

$$\models (\text{L1} \succ (\text{L2} \perp \text{L3})) \wedge \Diamond(\mathcal{P}(\text{L2}) \vee \mathcal{P}(\text{L3})) \supset$$
$$\exists e\, [(\text{HAPPENS}\ \mathcal{P}(\text{L1})?; e; \mathcal{P}(\text{L2})?) \vee (\text{HAPPENS}\ \mathcal{P}(\text{L1})?; e; \mathcal{P}(\text{L3})?)] \wedge (e \neq nil)$$

Proposition 8.4.

$$\models ((\text{L1} \perp \text{L2}) \succ \text{L3}) \wedge \Diamond\mathcal{P}(\text{L3}) \supset$$
$$\exists e\, [(\text{HAPPENS}\ \mathcal{P}(\text{L1})?; e; \mathcal{P}(\text{L3})?) \vee (\text{HAPPENS}\ \mathcal{P}(\text{L2})?; e; \mathcal{P}(\text{L3})?)] \wedge (e \neq nil)$$

Expressing a protocol family as a landmark expression. Using the definitions of partial order operator and the $\perp$ OR operator, the protocol family $\mathcal{F}$ in Figure 8.2 can be expressed by the following partial order expressions

$\text{L1} \succ (\text{L2} \perp \text{L4})$

$\text{L2} \succ \text{L3} \succ (\text{L5} \perp \text{L6})$

$\text{L4} \succ \text{L6}$

or more compactly as

$\mathcal{F} = \text{L1} \succ ((\text{L2} \succ \text{L3} \succ (\text{L5} \perp \text{L6})) \perp (\text{L4} \succ \text{L6}))$

This expression may be reduced to several equivalent forms, such as

$\mathcal{F} = (\text{L1} \succ \text{L2} \succ \text{L3} \succ (\text{L5} \perp \text{L6})) \perp (\text{L1} \succ \text{L4} \succ \text{L6}))$

We now look at landmarks in further detail.

8.2.2 Specializing, Generalizing, Realizing, and Instantiating Protocols

The landmarks in a landmark expression represent the 'way points' that must be followed (or the necessary states that must be achieved in the given order) to successfully execute the protocols belonging to the protocol family represented by that landmark expression. As such, the landmarks represent constraints on protocol execution. One can specialize a protocol family by introducing additional ordered landmarks and thus constraining the landmark-based representation further. Figure 8.3 represents a protocol family that is a specialization of the protocol family in Figure 8.2. The original ordering L1 $\succ$ L2 is still preserved by the new ordering L1 $\succ$ L7 $\succ$ L2 in the specialized protocol family.

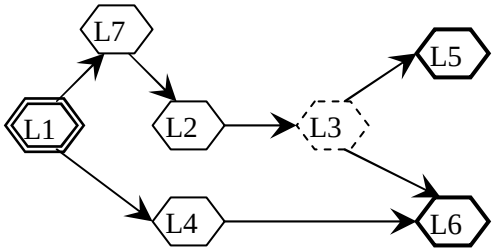


Figure 8.3: Specializing a protocol family

Similarly, one may generalize a protocol family by removing some of the landmarks and thus relaxing the constraints. When the landmarks are removed, the ordering between the remaining landmarks must be preserved. Figure 8.4 shows a protocol family that is a generalization of the protocol families in Figure 8.2 and Figure 8.3. The landmark L2 comes before L5 and L6 in the original protocol families as well as in the generalized protocol family.

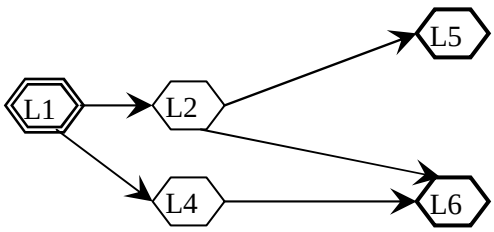


Figure 8.4: Generalizing a protocol family

The protocol families represented by landmarks can be reasoned about by intelligent agents and can be used for planning the communicative actions that need to be performed to successfully execute a protocol. But they are not of much use to agents who do not

possess reasoning and planning capabilities. The landmarks are used to represent protocol families and not concrete protocols. A concrete protocol is realized from a landmark-based representation of a protocol family by specifying action expressions for each landmark transition such that performing the action expressions provably results in the landmark transitions. Figure 8.5 represents a concrete protocol that is a realization of the protocol family in Figure 8.2.

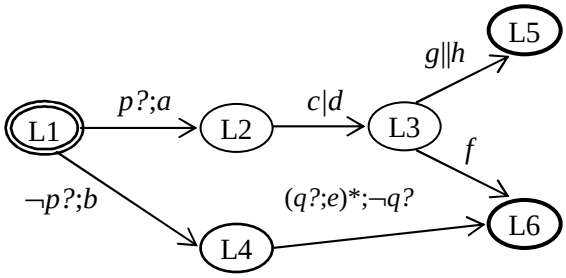


Figure 8.5: Realizing a protocol from a protocol family

We use circles to represent landmarks, and a different kind of arrows to represent landmark transitions, in order to distinguish concrete protocols from protocol families. In this figure, p and q are propositions and a, b, c, d, e, f, g, h are actions. These actions do not necessarily have to be communicative actions such as REQUEST and INFORM although we will analyze only communicative actions in this dissertation. Even though this concrete protocol looks similar to a finite state machine, there are several distinguishing features (1) the landmarks are precisely specified using propositions that are true in a landmark; (2) the landmarks are task oriented, that is, they are the waypoints towards achieving the goal of a protocol family and that goal must be achieved when the protocol ends properly; and (3) the landmark transitions can be due to arbitrarily complex action expressions consisting of communicative as well as other actions. One can represent concurrent performance of actions in this representation that was not possible in the finite state machine representation. Concrete protocols are represented logically as joint action expressions in the next section.

Though not shown explicitly, the actions in Figure 8.5 involve agent roles[3] such as initiator and participant of a protocol instead of agent instances. However, these roles must map to an agent or group instance in any actual execution of a conversation protocol. A

[3]A role refers to the abstract actor for a set of tasks, and an agent playing a role performs the tasks associated with that role. For example, an agent that performs the tasks that a personal secretary normally does, is said to be playing the role of a personal secretary.

concrete protocol is partially instantiated if there exists at least one agent role that is not yet mapped to an agent instance. A protocol is fully instantiated when all agent roles in that protocol map to some agent instantiation.

We want to apply the ideas introduced so far to well-known protocols. In order to do that, we will first introduce the logic needed to formally express the propositions that comprise landmarks and define the communicative actions that will be used to realize concrete protocols.

8.3 A WELL-KNOWN PROTOCOL FAMILY

We show how to derive the protocol family for some well-known protocols that require establishing a joint commitment to get an action done. We start with the classical state machine representation of the Request protocol and work backwards to find a set of partially ordered landmarks. Once we obtain the protocol family for the Request protocol, we generalize it and then prove that the Request protocol, the Standing Offer protocol and the Contract-Net protocol are realizations of this same family.

8.3.1 Request Conversation Protocol

Figure 8.6 shows a version of the popular "request for action" protocol adapted from the FIPA Request Interaction Protocol [41] using a finite state machine representation. The states are labeled as S1 to S6 and each state transition is labeled by the communicative act that results in that transition. The states S2, S4, S5, and S6 with dark outlines are possible final states of this protocol. An initiator x requests a participant y to do an action a. Participant y either refuses or agrees to the request. If the participant agrees to the request, the protocol terminates when either the initiator cancels its request or the participant informs that the action has been done or that it is impossible to do the action. Using the definitions of the communicative acts, it has been argued in [106, 107] that this protocol is used to get an action done by establishing a joint commitment between the initiator and the participant at S3.

We prove in Theorem 8.1 that our reformulation of this protocol does in fact establish a joint commitment between the initiator and the participant. This joint commitment can be discharged by establishing a mutual belief that either the action a is done, or is impossible or irrelevant. Both the initiator and the participant should be able to establish these mutual beliefs. However, there is no way in the state machine in Figure 8.1 for

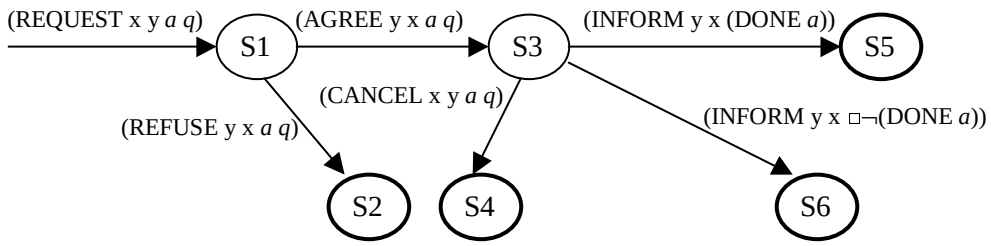
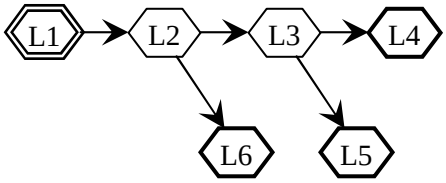


Figure 8.6: A Request Conversation Protocol as a Finite State Machine

the participant to establish mutual belief about irrelevance of the jointly committed goal. Without getting into the details about correctness and completeness of this protocol, we identify below the landmarks for the protocol family of this protocol (Figure 8.7). The arrows in Figure 8.7 indicate the ordering of the landmarks rather than state transitions as in Figure 8.6.



$\mathcal{F}_{\text{Request}}$ = L1≻ L2 ≻ (L6 ⊥ (L3 ≻ (L4 ⊥ L5)))

L1: ¬(DONE y *a*) ∧ ¬(JPG x y (DONE y *a*) (PWAG x y (DONE y *a*) *q*)∧*q*)
L2: (GOAL x ◊φ) ∧ (MB x y (PWAG x y φ *q*))
φ = (DONE y *a*) ∧ [PWAG y x (DONE y *a*) (PWAG x y (DONE y *a*) *q*)∧*q*]
L3: (JPG x y (DONE y *a*) (PWAG x y (DONE y *a*) *q*)∧*q*)
L4: (DONE y *a*) ∧ (MB x y (DONE y *a*))
L5: ¬(DONE y *a*) ∧ [(MB x y □¬(DONE y *a*)) ∨ (MB x y ¬*q*)
∨ (MB x y ¬(PWAG x y (DONE y *a*) *q*))]
L6: ¬(DONE y *a*)
∧ [(MB x y □¬(PWAG y x (DONE y *a*) (PWAG x y (DONE y *a*) *q*)∧*q*))
∨ (MB x y ¬(PWAG x y (DONE y *a*) *q*))]

Figure 8.7: Protocol Family For Request Protocol

When the protocol starts, the requested action has not been done by the initiator and the initiator and the participant do not have any joint commitment to do that action. Accordingly, we specify the first landmark L1 of this protocol.

L1: ¬(DONE y *a*) ∧ ¬(JPG x y (DONE y *a*) (PWAG x y (DONE y *a*) *q*)∧*q*)

By performing the request (REQUEST x y *e a q* t), the initiator has made public its goal and intention. The goal and intention are obtained using the definitions of REQUEST and ATTEMPT (Section 2.5). Assuming mutual belief establishment by default, from Theorem 2.2, the sender not only made its intention public by performing the request but also established mutual belief about its PWAG towards the requestee. These can be used

as the second landmark.

L2: (GOAL x $\Diamond\varphi$) $\wedge$ (MB x y (PWAG x y φ q))
where, φ = (DONE y a) $\wedge$ [PWAG y x (DONE y a) (PWAG x y (DONE y a) q)$\wedge q$]

If the participant agrees to the request, the agents have an interlocking PWAG, resulting in their having a joint commitment with each other (Theorem 2.1). This state gives us another landmark.

L3: (JPG x y (DONE y a) (PWAG x y (DONE y a) q)$\wedge q$)

There are two possible ending landmarks in each of which the joint commitments have been discharged. In one of the landmarks, the requested action will have been done, and in the other, either it was found to be impossible or irrelevant. Furthermore, the requester may cancel his request before the requested action has been done. These give the following two landmarks.

L4: (DONE y a) $\wedge$ (MB x y (DONE y a))

L5: $\neg$(DONE y a) $\wedge$ [(MB x y $\Box\neg$(DONE y a)) $\vee$ (MB x y $\neg q$)$\vee$
(MB x y $\neg$(PWAG x y (DONE y a) q))]

There will be no joint commitment if the participant refuses the initiator's request or if he does not respond by a specified deadline. In fact, by refusing, the participant has made it public that he will never have a commitment towards the initiator. Let us assume that the effect of the participant's not responding by the deadline is the same as that of the participant's performing the REFUSE communicative act. Moreover, the initiator can cancel his request, thereby, establishing mutual belief that he does not have PWAG towards the participant anymore. We can combine these to get another ending landmark.

L6 : $\neg$(DONE y a)$\wedge$
[(MB x y $\Box\neg$(PWAG y x (DONE y a) (PWAG x y (DONE y a) q)$\wedge q$)) $\vee$
(MB x y $\neg$(PWAG x y (DONE y a) q))]

Exceptions may lead to various landmarks with possibly un-discharged commitments. However, the exceptions are dealt with by an over-arching joint commitment, as we shall argue in the next section. The following partially ordered landmarks completely specify the protocol family of which the Request Conversation protocol is a concrete realization:
L1$\succ$ L2 $\succ$ L3 $\succ$ (L4 $\perp$ L5)

L2 $\succ$L6

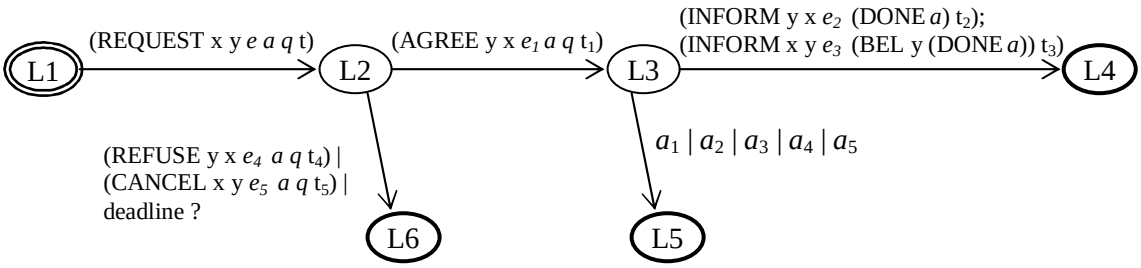
Or more compactly by

$\mathcal{F}_{Request}$ = L1$\succ$ L2 $\succ$ (L6 $\perp$ (L3 $\succ$ (L4 $\perp$ L5)))

This formula is represented visually in Figure 8.7.

Recall that a concrete protocol is realized from a landmark-based representation by specifying the landmark transitions using action expressions such that performing the actions provably results in the specified transitions. This yields the Request Conversation protocol in Figure 8.8. Using definitions of the communicative actions and the test action used in this protocol, one can prove that the action expressions do result in the appropriate landmark transitions. We illustrate one such example in Theorem 8.1 wherein we show that a REQUEST followed by an AGREE performed in landmark L1 results in transition from landmark L1 to L3. Note that sometimes a single communicative action such as CANCEL is sufficient to establish the mutual belief required by a landmark and sometimes an INFORM followed by a confirmation, that the informed proposition was believed by the receiver, is needed to achieve the same end. Whether or not a confirmation is required to establish mutual belief depends on the content of the communicative act as predicted by Lemma 2.2 and Lemma 2.3. The proposition deadline is defined as ($t_{current} > t_{start}+$ timeout) where t_{start} in this protocol is the time of performing the REQUEST action. The protocol in Figure 8.8 is a concrete protocol specifying the actions that result in landmark transitions and so it looks similar to the protocol specified using finite state machines in Figure 8.6. However, it is more precise than the state machine representation because the 'states' or the landmarks are formally defined, and therefore, can be used for logical reasoning. The most important difference between the landmark based representation in Figure 8.7 and the finite state machine based representation in Figure 8.6 is that the landmarks specify the crucial properties of protocol without regard to how the landmarks are reached. For example, informing that it is impossible to do the requested action a, as well as canceling the request, leads to the same landmark L5 from landmark L3. However, each of these actions may have very different consequences that the landmark-based representation does not care about – for example, canceling a request may result in the requester's having to pay a penalty but informing that the requested action is impossible may simply dissolve the 'contract'. We establish below that the communicative actions in the request protocol in Figure 8.8 do result in the specified landmark transitions. We show this for two different transitions, and one can similarly establish this result for all other landmark transitions in this protocol. The double turn-style arrow ($\Rightarrow$) in the

theorems in this chapter represents defeasible implication from Definition 2.11.



a_1 = (INFORM y x e_6 $\neg q$ t_6);(INFORM x y e_7 (BEL x $\neg q$) t_7)
a_2 = (INFORM y x e_8 $\Box\neg$(DONE a) t_8);(INFORM x y e_9 (BEL x $\Box\neg$(DONE a)) t_9)
a_3 = (CANCEL x y e_{10} a q t_{10})
a_4 = (INFORM x y e_{11} $\neg q$ t_{11});(INFORM y x e_{12} (BEL y $\neg q$) t_{12})
a_5 = (INFORM x y e_{13} $\Box\neg$(DONE a) t_{13});(INFORM y x e_{14} (BEL y $\Box\neg$(DONE a)) t_{14})

Figure 8.8: Realizing a Request Conversation Protocol From a Request Protocol Family

Theorem 8.1. *A REQUEST performed in landmark L1 results in landmark L2 and an AGREE performed in landmark L2 in response to the request results in landmark L3. The AGREE communicative act need not immediately follow the REQUEST communicative act provided that the propositions in landmark L2 are true when AGREE is performed and remain true at least until AGREE is done. Formally, we need to show two parts,*

$\models$ (DONE $\mathcal{P}$(L1)?;(REQUEST x y e a q t)) $\Rightarrow$ $\mathcal{P}$(L2)
$\models$ (DONE [$\mathcal{P}$(L2) $\wedge$ (UNTIL (DONE (AGREE y x e_1 a q t_1)) $\mathcal{P}$(L2))]?; e_1) $\Rightarrow$ $\mathcal{P}$(L3)

where, $\mathcal{P}$(Ln) is the conjunction of propositions true in landmark Ln

Proof. Part 1:

1. From Theorem 2.2, successful performance of the REQUEST communicative act establishes mutual belief by default between the requester and the requestee about the requester's PWAG towards the requestee. Therefore, we have (MB x y (PWAG x y φ q)), where

   φ = (DONE y a) $\wedge$ [PWAG y x (DONE y a) (PWAG x y (DONE y a) q)$\wedge q$]

2. By performing the request, the requester makes public its goal that $\Diamond\varphi$. This goal is obtained by using the definitions of REQUEST and ATTEMPT. By the sincerity assumption, the requester must have that goal, that is, (GOAL x $\Diamond\varphi$).

Therefore, successful performance of a REQUEST communicative act in landmark L1 results in landmark L2.

Part 2:

1. From Theorem 2.4, successful performance of an AGREE establishes mutual belief by default between the sender and the recipient about the sender's PWAG towards the recipient. Therefore, we have

   (MB x y (PWAG y x (DONE y a) (PWAG x y (DONE y a) q)$\wedge q$))

2. It is given that the propositions in landmark L2 are true when AGREE is performed. Therefore, from L2, we have, (MB x y (PWAG x y φ q)) where φ is given above.

3. Note that, for any propositions α,β, and δ,

   $\models$ (MB x y (PWAG x y $\alpha\wedge\beta$ δ)) $\wedge$ (MB x y β) $\supset$ (MB x y (PWAG x y α δ))

4. Using the first conjunct in φ, we have

   (MB x y (PWAG x y (DONE a) q)) [From 4, 5, and 3]

5. Therefore, using the theorem on inter-locking PWAGs (Theorem 2.1), we can conclude that

(JPG x y (DONE y a) (PWAG x y (DONE y a) q) $\wedge$ q) [From 3, 6, and Theorem 2.1]

This is $\mathcal{P}$(L3).

Therefore, we can conclude that the landmark L3 is achieve immediately after AGREE is performed.

□

A special case of this theorem is when REQUEST is immediately followed by an AGREE.

Theorem 8.2. *A REQUEST immediately followed by AGREE in response to the request establishes a joint persistent goal (JPG) between the initiator and the participant, assuming instantaneous communication. Formally,*

$\models$(DONE (REQUEST x y e a q t);(AGREE y x e_1 a q t_1))
$\Rightarrow$ (JPG x y (DONE y a) (PWAG x y (DONE y a) q)$\wedge q$)

Proof. 1. AGREE immediately follows the REQUEST and therefore, $\mathcal{P}$(L2) achieved by the REQUEST still holds when AGREE is performed. Moreover, $\mathcal{P}$(L2) must

remain true at least until AGREE is performed due to assumption of instantaneous communication.

2. Therefore, all conditions for Theorem 8.1 are satisfied. Hence, we can conclude that (JPG x y (DONE y a) (PWAG x y (DONE y a) q)$\wedge q$)

□

Next, generalize the request protocol family of Figure 8.7.

The protocol family in Figure 8.7 restricts the way in which joint commitment is established because one has to pass through landmark L2 before a JPG can be established and L2 requires that the JPG establishment process be initiated by the initiator role. Removing this restriction will result in a protocol family that is a generalization of the request protocol family. Landmark L2 provides an escape route from landmark L6 without establishing the JPG. Removing landmarks L2 and L6 from Figure 8.7 gives a family of protocols for getting an action done by establishing JPG between the initiator and the participant. Note that this protocol family does not indicate which role initiates establishment of the JPG. Therefore, either participant may initiate the process of establishing the JPG. This generalized family of protocol can be represented logically by the following landmark expression L1$\succ$ L3 $\succ$ (L4 $\perp$ L5) and is shown in Figure 8.9.

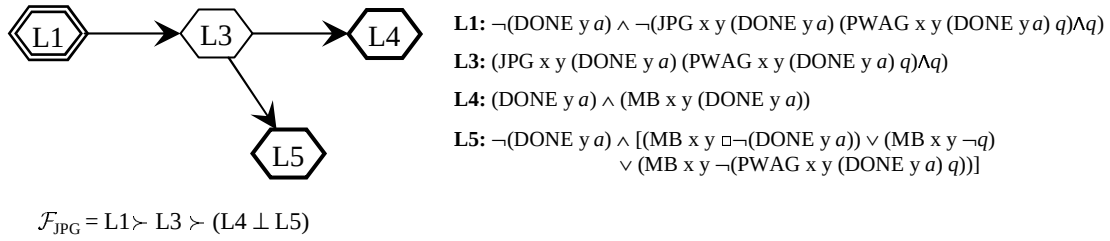


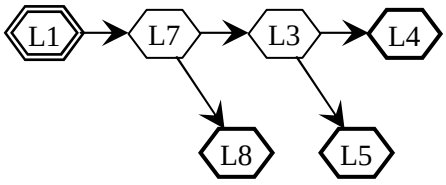
Figure 8.9: Family of Protocols for getting an action done by establishing JPG

The protocol family in Figure 8.9 represents a large number of protocols that establish and discharge a team while achieving a task. For instance, besides the Request Conversation protocol discussed above, it also represents the Standing Offer [107, 8], and the Contract-Net Conversation protocol because the only difference between these three protocols is the way in which a team is established.

8.3.2 Standing Offer Conversation Protocol

A Standing Offer conversation protocol belongs to the same family as the Request conversation protocol because both these protocols first establish a joint commitment and

then discharge it. We have seen that in a Request protocol, the JPG is established when the initiator requests the participant to do an action and the participant agrees to the request. The Standing Offer protocol in Figure 8.6 establishes JPG when the initiator makes a standing offer to do an action, and the other participant informs that he has the mental state required to accept the standing offer. Note that in case of Request, the initiator (i.e., the requester) is not the intended actor of the requested action, whereas in Standing Offer, the initiator is the intended actor of the action being offered. The Standing Offer conversation protocol family using landmark expressions is shown in Figure 8.10.



$\mathcal{F}_{\text{SOffer}}$ = L1≻ L7 ≻ (L8 ⊥ (L3 ≻ (L4 ⊥ L5)))

L1: ¬(DONE y *a*) ∧ ¬(JPG x y (DONE y *a*) (PWAG x y (DONE y *a*) *q*)∧*q*)
L7: (MB x y (BEL y φ))
φ = ∀ *e1* (DONE (INFORM x y *e1* (PWAG x y (DONE y *a*) *q*))) ⊃
(DONE *e1*;(PWAG y x (DONE y *a*) (PWAG x y (DONE y *a*) *q*)∧*q*)?)
L3: (JPG x y (DONE y *a*) (PWAG x y (DONE y *a*) *q*)∧*q*)
L4: (DONE y *a*) ∧ (MB x y (DONE y *a*))
L5: ¬(DONE y *a*) ∧ [(MB x y □¬(DONE y *a*)) ∨ (MB x y ¬*q*)
∨ (MB x y ¬(PWAG x y (DONE y *a*) *q*))]
L8: ¬(DONE y *a*)
∧ [(MB x y □¬(PWAG x y (DONE y *a*) *q*)) ∨ (MB x y (BEL y ¬φ))]

Figure 8.10: Protocol Family for the Standing Offer Conversation Protocol

A concrete realization of the protocol family in Figure 8.10 is shown in Figure 8.11. The concrete protocol uses SOFFER, ACCEPT, and REJECT communicative acts that we define next. These communicative acts are defined as compositions of the basic communicative acts that happen to be INFORM and REQUEST in our framework. We show in Theorem 8.3 that an SOFFER followed by an ACCEPT by the recipient of the SOFFER is sufficient to establish a JPG between the two agents by default.

Definition 8.3. *Standing Offer*

(SOFFER x y e a q t) ≜ (INFORM x y e φ t)

where, φ is

$\forall e1$ (DONE (INFORM y x $e1$ (PWAG y x (DONE x a) q))) ⊃
(DONE $e1$;(PWAG x y (DONE x a) (PWAG y x (DONE x a) q)∧q)?)

A standing offer from x to y to do an action a relative to q is a conditional offer that if ever y informs x that he has a PWAG that x does a then x will have a PWAG to do a relative to y's PWAG as well as relative to q. An SOFFER extends over time – the offering agent is agreeing to perform the action if the listening agent ever makes it known that he has the appropriate PWAG. An SOFFER does not commit the agent making the SOFFER towards the recipient of the SOFFER. So even if the sender x discovers after making the

SOFFER that he is no longer able to honor the SOFFER, he is not required to inform the agent y that the standing offer has been withdrawn. However, we show in Theorem 8.3 that acceptance of the SOFFER by agent y does establish mutual belief in interlocking PWAGs towards each other, resulting in a joint commitment between the two agents. Once the SOFFER JPG is established, it is immaterial if agent x withdraws the SOFFER because it is now bound by its PWAG towards agent y. A variation of the standing offer conversation protocol argues that simply accepting an SOFFER does not create a joint commitment between the two agents because the SOFFER may have been withdrawn but the other agent was not notified of the withdrawal. As such, a further confirmation is required from the agent who made the SOFFER, using the AGREE communicative act, for a joint commitment to be created. A discussion on this variation of the standing offer conversation protocol can be found in [107].

Definition 8.4. *Withdraw*

(WITHDRAW x y e a q t) $\triangleq$ (EARLIER φ)?;(INFORM x y e $\neg\varphi$ t)

where, φ is

$\forall e1$ (DONE (INFORM y x $e1$ (PWAG y x (DONE x a) q))) $\supset$

(DONE $e1$;(PWAG x y (DONE x a) (PWAG y x (DONE x a) q)$\wedge q$)?)

A WITHDRAW is a way for the agent who made the SOFFER to get out of the standing offer by saying that the implication in SOFFER no longer holds. It is defined as an INFORM that is performed in the context of an earlier standing offer.

Definition 8.5. *Accept*

(ACCEPT x y e a q t) $\triangleq$ (INFORM x y e φ t)

where φ = (PWAG x y (DONE y a) q)

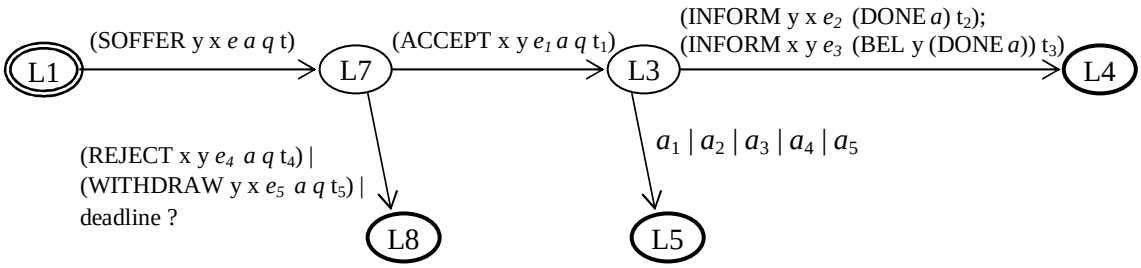
An accepting agent x informs the listening agent y that he (the accepting agent) has a PWAG with respect to y that y perform an action a and this PWAG is relative to some higher-level condition q. The main difference between the ACCEPT and the AGREE communicative acts is the relativizing condition of the sender's PWAG - in AGREE, the sender's PWAG is relative to the listener's PWAG (as well as relative to q) whereas in ACCEPT, it is relative only to q.

Definition 8.6. *Reject*

(REJECT x y e a q t) $\triangleq$ (INFORM x y e φ t)

where φ = ($\Box\neg$(PWAG x y (DONE y a) q))

The REJECT communicative act is the opposite of the ACCEPT communicative act - the rejecting agent x informs the listening agent y that x will never have the PWAG with respect to the listening agent that the listening agent does the specified action a. As in the case of ACCEPT and AGREE, the main difference between the REJECT and the REFUSE communicative act is the relativizing condition. The proposition deadline is defined as ($t_{current} > t_{start}$+ timeout) where t_{start}in this protocol is the time of performing the SOFFER action. We assume that the effect of the participant's not responding by the deadline is the same as that of the participant's performing the REJECT communicative act. Given these definitions, we are now in a position to prove that the communicative acts in Figure 8.11 do actually result in the landmarks specified in Figure 8.11.



a_1 = (INFORM y x e_6 $\neg q$ t_6);(INFORM x y e_7 (BEL x $\neg q$) t_7)
a_2 = (INFORM y x e_8 $\Box\neg$(DONE a) t_8);(INFORM x y e_9 (BEL x $\Box\neg$(DONE a)) t_9)
a_3 = (CANCEL x y e_{10} a q t_{10})
a_4 = (INFORM x y e_{11} $\neg q$ t_{11});(INFORM y x e_{12} (BEL y $\neg q$) t_{12})
a_5 = (INFORM x y e_{13} $\Box\neg$(DONE a) t_{13});(INFORM y x e_{14} (BEL y $\Box\neg$(DONE a)) t_{14})

Figure 8.11: A Standing Offer Conversation Protocol

As an example, we will prove that a joint commitment is created between the initiator and the participant as required by landmark L3.

Theorem 8.3. *An SOFFER performed in landmark L1 results in landmark L7 and an ACCEPT performed in landmark L7 in response to the SOFFER results in landmark L3. The ACCEPT communicative act need not immediately follow the SOFFER communicative act provided that the propositions in landmark L7 are true when ACCEPT is performed and remain true at least until the ACCEPT is done. Formally, we need to show two parts,*

$\models$ (DONE $\mathcal{P}$(L1)?;(SOFFER y x e a q t)) $\Rightarrow$ $\mathcal{P}$(L7)
$\models$ (DONE [$\mathcal{P}$(L7) $\wedge$ (UNTIL (DONE (ACCEPT x y e_1 a q t_1)) $\mathcal{P}$(L7))]?;e_1) $\Rightarrow$ $\mathcal{P}$(L3)
where, $\mathcal{P}$(Ln) is the conjunction of propositions true in landmark Ln,

Proof. Part 1:

1. SOFFER is defined as an INFORM that φ. From Theorem 2.3, successful performance of the INFORM communicative act establishes mutual belief by default between the agents that the sender believes φ. Therefore, (MB x y (BEL y φ)) where,

   φ = $\forall$ $e1$ (DONE (INFORM x y $e1$ (PWAG x y (DONE y a) q))) $\supset$

   (DONE $e1$;(PWAG y x (DONE y a) (PWAG x y (DONE y a) q)$\wedge q$)?)

   This is $\mathcal{P}$(L7) and therefore, successful performance of an SOFFER communicative act in landmark L1 results in landmark L7.

   Part 2:

2. ACCEPT is defined as an INFORM that (PWAG x y (DONE y a) q).

3. $\models$ (BEL x (PWAG x y (DONE y a) q)) $\supset$ (PWAG x y (DONE y a) q)

   [From Lemma 2.1]

4. Therefore, from 3, we can conclude that

   (MB x y (BEL x (PWAG x y (DONE y a) q)) $\supset$ (PWAG x y (DONE y a) q))

   [If φ is valid in our model, then it is mutually believed that φ]

5. Hence, we can conclude that

   (MB x y (PWAG x y (DONE y a) q)) [From 2, 4, and Lemma 2.2]

6. After ACCEPT has been performed, using definition of ACCEPT as an INFORM, we can conclude that

   (MB x y (DONE (INFORM y x $e1$ (PWAG y x (DONE x a) q))))

   [From Proposition 2.3]

7. From 6, we have

   (MB x y (BEL y (DONE (INFORM y x $e1$ (PWAG y x (DONE x a) q)))))

   [Using $\models$ (MB x y p) $\supset$ (MB x y (BEL y p))]

8. From 1 and 7, we can conclude that

   (MB x y (BEL y (PWAG y x (DONE y a) (PWAG x y (DONE y a) q)$\wedge q$)))

   [Using Modus Ponens]

9. Therefore, we can conclude that

   (MB x y (PWAG y x (DONE y a) (PWAG x y (DONE y a) q)$\wedge q$))

   [From 8 & Lemma 2.1]

10. From 5 and 9, and using Theorem 2.1 on inter-locking PWAGs, we can conclude that

    (JPG x y (DONE y a) (PWAG x y (DONE y a) q)$\wedge q$)

    This is $\mathcal{P}$(L3).

Therefore, we can conclude that the landmark L3 is achieve immediately after ACCEPT is performed.

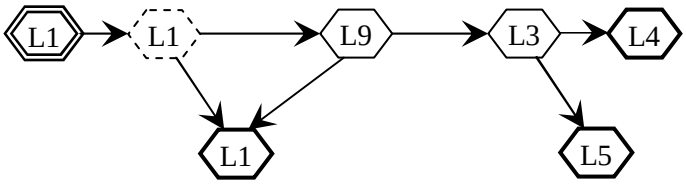
□

The Contract-Net protocol is similar to and belongs to the same protocol family as the Request and Standing Offer protocols in that it establishes a JPG between the initiator and the participant to get a task done. We discuss the Contract-Net protocol next.

8.3.3 Contract-Net Conversation Protocol

The Contract-Net protocol usually starts by the initiator's making a call-for-proposal, followed by a proposal from the participant, which is then accepted by the initiator. The Contract-Net protocol in Figure 8.13 is a simplified version of the FIPA Contract-Net protocol involving only one participant. The simplification made here does not affect the basic property of the Contract-Net protocol that a joint commitment is established between the initiator(s) and the participant(s) using the sequence of speech acts described here. The protocol family for this Contract-net protocol is shown in Figure 8.12. This protocol family is specialized from the Request protocol family in Figure 8.9 by adding additional landmarks L9, L10, and L11.

This Contract-Net protocol is based on the call for proposal (CFP), and the PROPOSE communicative acts. A call for proposal (CFP x y e a φ q t) is a solicitation for proposal from x to y that y propose to x a proposition φ such that if x has a commitment (PWAG) towards y relative to φ for doing an action a, then y will have a matching commitment towards x relative to x's commitment. A CFP may be composed as a REQUEST to INFORM the required proposition φ.



$\mathcal{F}_{\text{Contract-Net}}$ = L1$\succ$ L10 $\succ$ (L11 $\perp$ (L9 $\succ$ (L11 $\perp$ (L3 $\succ$ (L4 $\perp$ L5)))))

L1: ¬(DONE y *a*) ∧ ¬(JPG x y (DONE y *a*) (PWAG x y (DONE y *a*) φ)∧φ)

L10: (MB x y (PWAG x y (DONE (PROPOSE y x *e a* φ *q* t)) Q))

L9: (MB x y (PWAG y x ψ *q*))

ψ = ∃*e1* (DONE (INFORM x y *e1* (PWAG x y (DONE y *a*) φ))) ⊃
(DONE *e1*;(PWAG y x (DONE y *a*) (PWAG x y (DONE y *a*) φ)∧φ)?)

L3: (JPG x y (DONE y *a*) (PWAG x y (DONE y *a*) φ)∧φ)

L4: (DONE y *a*) ∧ (MB x y (DONE y *a*))

L5: ¬(DONE y *a*) ∧ [(MB x y □¬(DONE y *a*)) ∨ (MB x y ¬φ) ∨ (MB x y ¬(PWAG x y (DONE y *a*) φ))]

L11: ¬(DONE y *a*) ∧ [(MB x y □¬(PWAG x y (DONE y *a*) φ)) ∨ (MB x y ¬(PWAG y x ψ *q*))]

Figure 8.12: Protocol Family for the Contract-Net Protocol

According to this composed definition, a CFP is satisfied (i.e., the requested action gets done) when a PROPOSE is performed in response to the CFP. Even though a CFP can be used to elicit a PROPOSE, there is no reason why an agent cannot propose on its own. The crucial part of a Contract-Net protocol is the PROPOSE communicative act and not the call for proposal. We show in Theorem 8.4 that a PROPOSE followed by an ACCEPT in response to the PROPOSE creates a joint commitment between the proposer and the acceptor, and therefore, we need to formally define the PROPOSE communicative act. Also, the proposition deadline1 in Figure 8.13 is defined as ($t_{current}$ > t_{start} + timeout1) where t_{start}in this protocol is the time when the CFP was performed, and deadline2 is defined as ($t_{current}$ > t_1 + timeout2) where t_1 is the time when the PROPOSE communicative action was performed.

Definition 8.7. *Propose*

(PROPOSE x y e a φ q t) $\triangleq$ (INFORM x y e φ t)

where,

φ = (PWAG x y ψ q), and

ψ = $\forall$e1 (DONE (INFORM y x e1 (PWAG y x (DONE x a) φ))) $\supset$
(DONE e1;(PWAG x y (DONE x a) (PWAG y x (DONE x a) φ)$\wedge\varphi$)?)

A PROPOSE from x to y is an INFORM that the sender x has a PWAG with respect

to the recipient y about an implication ψ and this PWAG is relative to some higher-level condition q. The implication says that if y informs x that he has a PWAG towards x relative to a φ then x will also have the matching PWAG towards y relative to y's PWAG and relative to the condition φ. Assuming that the proposing agent is sincere, this definition requires the proposing agent to have a PWAG towards an implication, thus ensuring that if for some reason, the proposing agent decides that his proposal is no longer valid then he is bound by the PWAG (towards the implication) to establish mutual belief about this fact. A consequence of this fact is that a team is formed, that is, a JPG is established between x and y when y accepts the proposal using the ACCEPT communicative act. We state this result formally in Theorem 8.4.

Theorem 8.4. *A PROPOSE followed by a subsequent ACCEPT of the proposal establishes a JPG between the initiator and the participant. Formally,*

$\models$ (DONE [$\mathcal{P}$(L9) $\wedge$ (UNTIL (DONE (ACCEPT x y e_2 a q t_2)) $\mathcal{P}$(L9))]?;e_2) $\Rightarrow$ $\mathcal{P}$(L3)
where, $\mathcal{P}$(Ln) is the conjunction of propositions true in landmark Ln

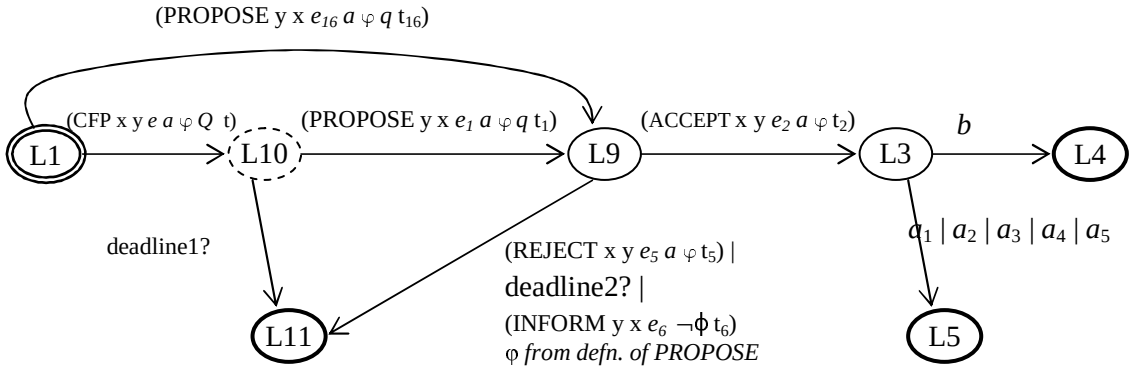
Proof. 1. As in proof of Theorem 8.1 and Theorem 8.3, using the definitions of the communicative acts PROPOSE and ACCEPT (Definitions 8.7 and 8.5), we first establish that there is a mutual belief in each other's PWAGs. The acceptor's PWAG is relative to the condition φ and the proposer's PWAG is relative to the acceptor's PWAG and the condition φ.

2. Thereafter, the desired result follows by application of Theorem 2.1.

□

The Contract-Net protocol in Figure 8.13 specifies an optional landmark. There are two ways to get from L1 to L9 – either through L10 or directly from L1 to L9. In one case, the proposing agent proactively makes a proposal and in the other case, it proposes in response to a call for proposal. Assuming, that a CFP is defined as a REQUEST, performing a CFP makes public the PWAG of the agent x making the call for the proposal. This PWAG is towards agent y for performing an INFORM and it gets discharged when agent y performs a PROPOSE (which is an INFORM) in response to the CFP. Therefore, the only relevant commitments in landmark L9 are those introduced by performing the PROPOSE and hence landmark L10 is an optional landmark. A 'proposer' agent in a community of intelligent agents all of whom plan their communicative actions using landmarks, can skip landmark L10 at execution time by proactively performing a PROPOSE

without waiting for a CFP if unsolicited proposals are allowed.



b = (INFORM y x e_3 (DONE a) t_3);(INFORM x y e_4 (BEL y (DONE a)) t_4)

a_1 = (INFORM y x e_7 $\neg\varphi$ t_7);(INFORM x y e_8 (BEL x $\neg\varphi$) t_8)

a_2 = (INFORM y x e_9 $\Box\neg$(DONE a) t_9);(INFORM x y e_{10} (BEL x $\Box\neg$(DONE a)) t_{10})

a_3 = (CANCEL x y e_{11} a φ t_{11})

a_4 = (INFORM x y e_{12} $\neg\varphi$ t_{12});(INFORM y x e_{13} (BEL y $\neg\varphi$) t_{13})

a_5 = (INFORM x y e_{14} $\Box\neg$(DONE a) t_{14});(INFORM y x e_{15} (BEL y $\Box\neg$(DONE a)) t_{15})

Figure 8.13: A Contract-Net Conversation Protocol

Given the family of protocols L1$\succ$ L3 $\succ$ (L4 $\perp$ L5) in Figure 8.9, one can systematically derive the concrete protocols such as the Request, the Standing Offer, and the Contract-Net protocols by first specializing this protocol family by inserting additional landmarks for the way in which joint commitment is to be established, and then specifying communicative and other action expressions that result in landmark transitions. An intelligent reasoning and planning agent would need to know only the definitions of protocol families using landmarks, definitions of allowed communicative acts, and whether or not there is joint commitment towards a protocol family for it to be able to generate concrete conversation protocols.

Next, we explore the consequences of applying the joint intention theory to protocol families and concrete protocols.

8.4 APPLYING JOINT INTENTION THEORY TO PROTOCOLS

We first argue for joint commitment towards a protocol family and explore the consequences of such a joint commitment. Thereafter, we explore the consequences of jointly

intending the action expression representation of a protocol.

8.4.1 Joint Commitment Towards a Protocol Family

Human society has developed the equivalent of a set of commonly understood conversation protocols for many common interactions. For example, if I ask you a question, you will either answer the question, or let me know if you do not know the answer, or ask for clarification if you did not properly understand my question, or direct me to who you think might be able to answer my question or where I might be able to find an answer. In fact, every person who abides by human social norms will do the same on realizing that he is a participant of a question-answer protocol. Just the fact that the question-answer protocol is common knowledge in our society is not sufficient to explain this conversation. One can, of course, speculate that everybody in the society is helpful and this fact along with common knowledge will explain the question-answer conversation. However, there is a better and much more useful hypothesis – the assumption that it is a social norm that we are jointly committed to the question-answer protocol offers a straightforward explanation to the question-answer conversation. You recognize that the question-answer protocol is being used and instantiate the initiator role with the person asking the question and the participant role with yourself. You also realize that as a result of the social norm, there is an over-arching joint commitment between the initiator and yourself and therefore, you discharge the joint commitment by communicating to establish the appropriate mutual beliefs.

One can identify several such instances where we find ourselves jointly committed to a protocol because of the governing norms [18] of the society, the social institutions, and other institutions that we might be part of. These joint commitments come into effect when a protocol gets instantiated. However, pre-existing societal norms do not cover all conversation protocols; there are also protocols to which we explicitly commit – for instance, two businesses jointly commit to a particular bill-payment protocol when they sign a trade agreement. Whether the joint commitment between the initiator and the participant of a protocol is provided by the governing social norms or by explicit contract, such over-arching joint commitments lead to proper communication, robust protocol execution by handling of exceptional situations, correctness criteria for protocols, and possibly dynamic realization of concrete protocols from protocol families. We express protocol families precisely as landmark expressions in a propositional dynamic logic and therefore, we can apply the joint commitment operator (JPG) directly to protocol families.

A number of important properties and behavior can be shown to hold when an over-arching joint commitment exists. Consider the family of protocols $\mathcal{F}$ for getting a task done by forming a joint commitment. From Section 8.3.1,

$\mathcal{F}$ = L1$\succ$ L3 $\succ$ (L4 $\perp$ L5)

Suppose that there is an over-arching joint commitment towards this protocol family (JPG x y $\mathcal{F}$ q) in the context of a particular social institution. The relativizing condition q then refers to this social context which is the membership of the initiator x and participant y in the social institution at hand. Using the definition of JPG (Definition 2.6) and the landmark ordering and OR operators from Section 8.2.1, one can predict the following behavior and properties.

- Appropriate Communication. JPG specifies that a jointly committed goal will persist until there is mutual belief among the agents involved about its achievement, impossibility or irrelevance. If an agent privately comes to believe that the jointly committed goal has been achieved or is impossible or irrelevant then it will have an individual commitment to bring mutual belief about the privately discovered fact. Establishing mutual belief requires communication of some sort – either by exchanging explicit messages or by mutually understood signaling or by some other means. Therefore, joint commitment towards a protocol family predicts that there will be appropriate communication to establish the required mutual beliefs among the agents involved in any protocol that realizes a protocol family. The initiator and the participant will eventually mutually believe that the protocol execution has been successfully achieved or was impossible or irrelevant. The initiator and the participant of a protocol get jointly committed to that protocol by (1) a mutual belief that there is an over-arching joint commitment towards the protocol family of which the protocol at hand is an instance, and (2) a mutual belief or recognition that they (i.e., the agents) instantiate the roles involved in the protocol.

- Automatic Exception Handling. Joint commitment characterizes teamwork and agents bound together by joint commitment form a robust team that can handle exceptions, failures, and adverse situations. The persistence of the jointly committed goal and the requirement to establish mutual belief ensures that agents can depend on each other. If something goes wrong the agents will attempt to resolve it by communicating to establish mutual belief. For instance, if an agent does not understand a message, it will communicate this fact to the sender of the message because of

the joint commitment. As such, performatives such as "not understood" and "retransmit" that tend to clutter traditional protocol diagrams are taken care of as a consequence of jointly committing to a protocol. Therefore, such exceptions related to execution of a protocol need not be specified as part of the protocol.

- Correctness Criterion. Landmarks are the waypoints towards achieving the goal that a protocol or protocol family is meant for. Therefore, the main correctness criterion for any protocol is whether or not successful execution of that protocol achieves its goal. The goal of a protocol is specified as a proposition that is true only in the final landmark. If there are multiple final landmarks, the goal is true in the 'main' final landmark that represents the successful execution of all protocols in that protocol family. For example, the goal of the protocol family $\mathcal{F}$ in our earlier examples is (DONE y a) where y is the participant role. A protocol family specified using landmarks might be used by an intelligent agent with reasoning and planning capabilities to figure out the communicative acts required to achieve the goal of that protocol family. However, most agents that lack these capabilities do not care about the landmarks – they need to know the complete communicative action expression that can achieve the required goal. These two cases represent the two possible extremes – on one hand we have landmarks that may consist of mental states internal to an agent and on the other hand we have observable and executable communicative actions. One can think of landmark-based representation as a protocol design specification and a concrete protocol as a protocol implementation. Given a concrete protocol that is completely specified using communicative actions, the assumption of joint commitment towards a protocol family gives a straightforward criterion to determine whether or not the protocol is correct – a joint intention towards the action expression representation of a protocol must satisfy the joint commitment towards the landmark expression for that protocol. Further, there can be several intermediate levels of protocol representation between the two extreme representations of protocols, for instance, partially realized protocol families that specify a few landmarks along with a few communicative acts. Each level of representation gives the correctness criterion for the next more concrete level – the JPG towards a protocol represented at a certain level must satisfy the JPG towards the same protocol represented at the previous level. Of course, an action expression for a protocol must also satisfy the landmark constraints for the protocol family of that protocol in order for the protocol to be correct with respect to its family.

We illustrate this idea with our running example of the protocol family $\mathcal{F}$. We want to determine whether an action expression $a1;a2;a3;a4$ where $a1$, $a2$, $a3$, $a4$ are communicative acts, correctly specifies a concrete protocol belonging to this family. According to our correctness criterion, we want to know whether satisfying the joint intention (JI x y $a1;a2;a3;a4$ q) also counts as satisfying the joint commitment (JPG x y φ q) where φ is a proposition involving landmarks defined earlier. The problem reduces to the following two criterion.

1. It should be possible to execute the action expression $a1;a2;a3;a4$ in the starting landmark Ls. Also, successful execution of this action expression in landmark L1 must result in a final landmark (L4 | L5). Combining these two cases, it should be possible to successfully execute the action expression $\mathcal{P}$(L1)?;$a1;a2;a3;a4$;($\mathcal{P}$(L4) $\vee$ $\mathcal{P}$(L5))?

2. It should be possible to identify the intermediate landmark L3 after completion of one of the actions in the action expression. In other words, it should be possible to successfully execute one of the following action expressions:
$\mathcal{P}$(L1)?;$a1$;$\mathcal{P}$(L3)?;$a2;a3;a4$;($\mathcal{P}$(L4) $\vee$ $\mathcal{P}$(L5))? or $\mathcal{P}$(L1)?;$a1;a2$;$\mathcal{P}$(L3)?;$a3;a4$;($\mathcal{P}$(L4) $\vee$ $\mathcal{P}$(L5))? or $\mathcal{P}$(L1)?;$a1;a2;a3$;$\mathcal{P}$(L3)?;$a4$;($\mathcal{P}$(L4) $\vee$ $\mathcal{P}$(L5))?

Concrete protocols are realized from a landmark-based representation of protocol families using action expressions for landmark transitions. Conversation protocols are, therefore, patterns of communicative action expressions. As such, they will be amenable to representation as action expressions using constructs of dynamic logic. Next, we discuss the consequences of jointly intending concrete protocols represented as action expressions.

8.4.2 Jointly Intending a Conversation Protocol

We can assume an over-arching joint intention (JI) towards concrete protocols by similar arguments as for an over-arching JPG towards protocol families in the previous section. The JI towards a protocol can result from either an institutional norm or an explicit contract. Recall that agents x and y jointly intend to do an action (or actions) a if they have a joint commitment to do the action a mutually believing throughout the action execution that they are jointly doing that action (Definition 2.7).

A concrete conversation protocol can be represented as a joint action expression and therefore, the joint intention operator can be applied to protocols. We analyze (JI x y Π q)

and predict at least the following consequences of jointly intending a protocol Π using the analyses of JI in [70, 29]. These consequences are what can be expected when STAPLE agents execute conversation protocols represented as joint actions.

- The definition of JI requires that there be a mutual belief throughout the protocol execution that the agents are jointly executing the protocol Π. Therefore, communication of some sort is required to establish the starting mutual belief. The first communicative act in a protocol may be used to signal the start of a protocol by specifying the protocol being used as part of the relativizing condition of the communicative act. If so, then the starting mutual belief can be established by simply performing the first communicative in a protocol. This may be implemented by indicating the protocol being used as part of the first message being sent during protocol execution. Furthermore, if there is any chance that the initial mutual belief may dissipate over the duration of protocol execution, then the definition of JI predicts intermediate "reassuring" communication to maintain the starting mutual belief. Of course, this can again be opportunistically achieved to some extent by specifying the protocol as part of the relativizing condition of the communicative acts.

- As a result of the joint intention, both agents are committed not only to their part of the protocol but also to the entire protocol. Therefore, they will not intentionally do something to render the performance of communicative acts by the other agent impossible, thereby making the protocol execution impossible. In fact, each agent's commitment towards the other agent's communicative acts may lead to the agent's helping each other in performance of their communicative acts. Executing a jointly intended protocol in a noisy environment, for instance stock trading on the floor of a stock exchange using jointly agreed upon trading protocols, is a situation where this behavior predicted by joint intention is very important.

- A joint intention towards a protocol leads to the appropriate individual intentions relative to the broader joint intention. For instance, if a protocol requires the agents to concurrently perform their respective communicative acts, then each agent will privately intend to perform its communicative act relative to the joint intention. If the agents jointly intend to execute the sequential parts of a protocol in lockstep [29] then they will have individual intentions to bring about appropriate mutual belief after executing each sequential step of the protocol [29].

- JI is defined in terms of JPG. The definition of JPG requires that there be mutual belief among the participants of a protocol about the completion, impossibility, and irrelevance of executing that protocol in order to discharge the JPG. Therefore, (JI x y Π q) predicts that there will be communication among the participants of a jointly intended protocol to establish the appropriate mutual belief about completion, impossibility or irrelevance of a protocol execution. Furthermore, we will represent protocols as joint action expressions in the next section. Since a protocol Π is nothing but an action expression, one can capture the fact that certain actions are being done as a result of jointly intending a protocol by using the term (DOING x y Π) in the relativizing term $q$of a joint intention formula.

- Joint intention towards a protocol predicts robust execution of that protocol. Exceptions are handled automatically because of the requirement to establish mutual belief due to the definition of JI as a JPG. If a participant of a protocol comes to privately believe that the protocol has been completed, or that the protocol is impossible to complete, or it is irrelevant to execute the protocol then he is committed to establishing the corresponding mutual belief. Communicative acts such as 'not-understood' as in KQML [67] and FIPA [41] need not be specified as part of the protocol because the agents will use the appropriate exception handling communicative acts when needed. This is a consequence of the persistence of joint intentions – if an agent doesn't understand a communicative act (i.e., if it is unable to deduce the effect, precondition, etc.) performed as part of a jointly intended protocol then it will ask for clarification etc. and try all possible means at its disposable before it can come to a conclusion that it is impossible to execute the jointly intended protocol. Even then, the agent must first establish mutual belief that the protocol execution is impossible before it can give up.

A consequence of using joint intention theory to analyze protocols is that it gives us correctness and completeness criterion for protocols as well as for their compositions. Next, we use the joint intention theory to first represent protocols as joint action expressions and then define and analyze protocol compositions.

8.5 COMPOSING PROTOCOLS

Concrete protocols are realized from a landmark-based representation of protocol families using action expressions for landmark transitions. Here, we propose an action expression

based representation for protocols, define different types of protocol compositions using that same representation, propose criterion for meaningful compositions, and apply these techniques by representing the Request protocol as an action expression composed of protocols to form and discharge teams.

8.5.1 Representing Concrete Protocols

We use the dynamic logic operators for action sequences ($a;b$), concurrent actions ($a||b$), non-deterministic OR ($a|b$), test action ($p?$), and indefinite repetitions ($a*$) where a and b are actions and p is a proposition. We also define and use a rational choice operator ($a \uparrow b$) specifying that the agent selects and executes one of the actions using some rational decision process.

We require an OR operator to represent that an agent must use its reasoning process to determine which alternative action it chooses to execute. For example, when an agent receives a request in a Request protocol, we want to say that the agent either refuses or accepts but do not want to specify how the agent decides what to choose. 'How to choose' is something internal to the agent – it may use utility based or any other criteria such as whether it has enough resources, whether it can commit, whether it conflicts with prior commitments etc. to determine what to choose. We introduce a *rational choice operator* $\uparrow$ and give it a semantics using the non-deterministic OR operator.

Definition 8.8. *Rational Choice.*

Let a and b be two actions having the same agent, that is, (AGT a) = (AGT b) and let Δ be an operator that determines the utility of doing an action for that agent. The rational choice for the agent of actions a and b is defined as

$a \uparrow b \triangleq (\Delta a > \Delta b)?; a|\ \neg(\Delta a > \Delta b)?; b$

Pragmatically, Δ represents some reasoning process internal to an agent. One may use results from decision and game theory literature to define the Δ operator.

Conversation protocols are applicable in certain contexts and are used to achieve certain goals. One can think of the starting landmark as specifying the precondition and the main final landmark as specifying the goal associated with the protocol. We incorporate the precondition and the goal associated with a conversation protocol in its representation as a joint action expression such that executing that joint action expression amounts to correct execution of the protocol using the main correctness criterion from Section 8.4. Accordingly, we view a joint action expression as having three components – a test to

determine whether the precondition is true, an action expression to achieve the goal associated with the protocol, and a test to know if that goal is achieved. For example, let p be the precondition, $a;(b|c)$ be the action expression, and let g be the goal of a protocol. Further, assume that $a;b$ is the main path in this protocol, that is, the path from the starting landmark to the main final landmark in the landmark-based representation of the protocol family to which this protocol belongs. Then this protocol $\Pi_{example}$ is specified in terms of joint action expressions as

$\Pi_{example} = p?; a; ((b; g?)|\mathrm{c})$

Just as a protocol family can have multiple final landmarks, a protocol may have multiple goals. For example, if the goals of the protocol in this example were $g1$ and $g2$ corresponding to the two actions b and c, then this protocol would be represented as

$\Pi_{example} = p?; a; ((b; g1?)|(c; g2?))$

Using this technique, the request for action protocol in Figure 8.8 can be expressed as:

$\mathcal{P}$(L1)?; c_1; [(c_4; $\mathcal{P}$(L6)?) |(c_2; [(c_5; $\mathcal{P}$(L5)?) |(c_3; $\mathcal{P}$(L4)?)])]

where,

c_1 = (REQUEST x y e a q t)

c_2 = (AGREE y x e_1 a q t_1)

c_3 = (INFORM y x e_2 (DONE a) t_2);(INFORM x y e_3 (BEL y (DONE a)) t_3)

c_4 = (REFUSE y x e_4 a q t_4) | (CANCEL x y e_5 a q t_5) | deadline?

c_5 = ((INFORM y x e_6 $\neg q$ t_6);(INFORM x y e_7 (BEL x $\neg q$) t_7))|

((INFORM y x e_8 $\Box\neg$(DONE a) t_8);(INFORM x y e_9 (BEL x $\Box\neg$(DONE a)) t_9))|

(CANCEL x y e_{10} a q t_{10}) |

((INFORM x y e_{11} $\neg q$ t_{11});(INFORM y x e_{12} (BEL y $\neg q$) t_{12}))|

((INFORM x y e_{13} $\Box\neg$(DONE a) t_{13});(INFORM y x e_{14} (BEL y $\Box\neg$(DONE a)) t_{14}))

Protocols are action expressions and therefore, (1) teams of agents can jointly intend protocols represented as action expressions, and (2) protocols can themselves be composed using the same operators used to compose action expressions. We first analyze the effects of jointly intending a conversation protocol and then look into issues involved in protocol composition.

8.5.2 Compositions

There are two main issues in composing protocols - what are the possible compositions of protocols and what are the correctness criteria for protocol compositions.

We have argued that protocols, and therefore, composed protocols can be regarded as action expressions. As such, protocols can be composed using the operators of dynamic logic of actions. If Π_1 and Π_2 are two protocols and p is a proposition, then $\Pi_1;\Pi_2$ (sequence), $\Pi_1|\Pi_2$ (non-deterministic or), $\Pi_1||\Pi_2$ (concurrent execution), $\Pi_1{}^*$ (repetition), and $p?;\Pi_2$ (conditional execution) and any combination of these is a possible syntactic composition subject to certain semantic constraints. Actions in an action expression may be high-level actions that can be replaced by the equivalent (sub)action expression. This gives another composition operator – for embedding or replacing an action expression in a protocol by another protocol. If Π and π are two protocols then $\Pi[a/\pi]$ represents the protocol obtained by substituting a (sub)action expression a in Π by the action expression for protocol π. Recall that the joint action expression representation of a protocol consists of a test for precondition, an action expression, and a test for goal achievement. The composition operators are first applied only on the action expression part, that is, after stripping off the precondition test and goal achievement tests for each protocol being composed. The precondition and goal for the composed protocol are then combined with the composed action expression to yield the complete composed joint action expression. The precondition of the composed protocol is derived from the precondition of the constituent actions.

Given the various possible ways to compose protocols, we want to determine which of the compositions are meaningful. One can identify at the least following criteria for correctness and legality of protocol compositions.

1. **Completeness and Ending criteria.** In the previous teamwork-based analysis of protocols [106, 107], we have argued that a complete conversation should not leave behind any un-discharged commitments. However, this completeness criterion is not applicable to protocols (either individual or composed) that have the goal to bring about certain commitments – for instance, a protocol to form a team is intended to leave behind a joint commitment among the participants. At most, we can say that any complete protocol of which the protocol to form a team is a component must have another protocol as its component in a sequence that results in discharging the joint commitment created by the previous (sub)protocol. Therefore, teamwork analysis gives us a *completeness criterion* – a criterion to determine whether a protocol is complete or partial and an *ending criterion* - a criterion to determine the acceptable end-points of a protocol.

2. **Enabling Criteria.** Conversation protocols consist of communicative acts that are themselves defined to bring about changes in states. Therefore, it seems intuitive that there must be some relation between the states at the end-point of one protocol and the starting point of another protocol. We call this the *enabling criterion* for composing protocols. For instance, the precondition of the successor protocol must be entailed by the effect of the prior protocol. Also, the commitments at the end point of one protocol (the output commitments) and at the starting-point of another protocol (the input commitments) must be related. The enabling criterion is best-expressed using landmarks. Consider for example, two protocols Π_1 and Π_2. Let Lf_1 be the final landmark of the protocol Π_1 and Ls_2 be the starting landmark of the protocol Π_2. The conjunction of propositions in the initial and the final landmarks of Π_1 are represented by $\mathcal{P}(\mathrm{Ls}_1)$ and $\mathcal{P}(\mathrm{Lf}_1)$ respectively and that of Π_2 by $\mathcal{P}(\mathrm{Ls}_2)$ and $\mathcal{P}(\mathrm{Lf}_2)$ respectively. A sequential composition $\Pi_1;\Pi_2$ is meaningful if the final landmark of Π_1 implies the starting landmark of Π_2. Formally,

$$\Pi_1;\Pi_2 \text{ is meaningful iff } \models \mathcal{P}(\mathrm{Lf}_1) \supset \mathcal{P}(\mathrm{Ls}_2)$$

Without this implication relationship, it is difficult to make guarantees about the resulting composition. This same enabling criterion is applicable to substitution (or embedding) composition $\Pi_1[a/\Pi_2]$ at the point of composition. The non-deterministic OR composition $\Pi_1|\Pi_2$ is meaningful if the starting landmarks of Π_1 and Π_2 entail each other. Formally,

$$\Pi_1|\Pi_2 \text{ is meaningful iff } \models \mathcal{P}(\mathrm{Ls}_1) \equiv \mathcal{P}(\mathrm{Ls}_2)$$

In other words, an OR composition of protocols is meaningful if the protocols being composed have the 'same' starting landmark, that is, the protocols being composed should be applicable in the same 'state'. A concurrent composition $\Pi_1||\Pi_2$ is meaningful if the starting and final landmarks of Π_1 are consistent with the starting and final landmarks respectively of Π_2. Formally,

$\Pi_1||\Pi_2$ is meaningful iff $(\mathcal{P}(\mathrm{Ls}_1) \wedge \mathcal{P}(\mathrm{Ls}_2))$ and $(\mathcal{P}(\mathrm{Lf}_1) \wedge \mathcal{P}(\mathrm{Lf}_2))$ are each jointly satisfiable.

Semantically, a concurrent composition of protocols has the same restrictions for the starting landmarks as an OR composition. In addition, when the concurrent protocols end, their end states should be consistent.

It is possible to define several other criterions for correctness and legality of protocol compositions. For instance, consider a composition $\Pi_1;\Pi_2;\Pi_3$ and assume that a certain proposition p is specified in the final landmark of Π_1 and in the starting landmark of Π_3

but is neither specified in nor entailed by the starting and ending landmarks of Π_2. If it can be shown by analysis of all actions in all the paths from starting landmark to the final landmark of Π_2 that the protocol Π_2 does not affect proposition p, then the composition $\Pi_1;\Pi_2;\Pi_3$ should be valid because all the propositions in the starting landmark of Π_3 will be entailed. However, we will not use any such criterion in the current work. We now apply the ideas introduced here to view the Request protocol as a composition of three different protocols and specify each (sub) protocol as well as their composition as joint action expressions.

8.5.3 Request Protocol as a Composition

Here, we will consider the Request protocol derived from the protocol family L1$\succ$ L3 $\succ$ (L4 $\perp$ L5) in Figure 8.9, as a composition of three different protocols – one to form a team between the initiator and the participant, and one each for the initiator and the participant to discharge their respective team commitments by establishing the appropriate mutual beliefs. We first formally specify each (sub) protocol as a joint action expression and then compose them to yield the Request protocol discussed in the previous section. The resulting composition is shown to be meaningful under the above composition criterion.

Protocol to establish a joint commitment to do an action.

The initiator x requests the participant y to do an action a with respect to some relativizing condition q, that is, the following communicative act is performed:

(REQUEST x y e a q t_s)

Before the deadline is over, the participant may either agree or refuse to do the requested action. Alternatively, the requester may cancel his request. Therefore, (AGREE y x e_1 a q t_1) or (REFUSE y x e_2 a q t_2) or (CANCEL x y e_3 a q t_3) may be performed. This protocol ends when either the deadline is reached or the participant has accepted or refused the request or the request has been cancelled. The goal or the purpose of this protocol is to establish joint commitment between the initiator and the participant for doing the requested action. Formally, this goal is given by

g = (JPG x y (DONE y a) (PWAG x y (DONE y a) q)$\wedge q$)

The precondition for this protocol is that the participant has not done the requested action and there is no joint commitment between the initiator and the participant for doing the requested action. Formally, the precondition p is given by

$p = \neg$(DONE y a) $\wedge$ $\neg$(JPG x y (DONE y a) (PWAG x y (DONE y a) q)$\wedge q$)

The deadline is given by the proposition ($t_{current} > t_s$ + timeout) where t_s is the time at which the protocol was started which is the time at which the REQUEST communicative act was performed. The participant must make a rational choice between agreeing to a request and refusing it depending on its internal reasoning process. The entire protocol can be represented as the following joint action expression.

$\Pi_{form-team}$(x, y, a, q, timeout) =

p?; request;{(¬deadline?;[(agree;g? ↑ refuse) | cancel]) | deadline?}

where,

p, g are the precondition and goal of the protocol as given above,

request = (REQUEST x y e a q t_s),

agree = (AGREE y x e_1 a q t_1),

refuse = (REFUSE y x e_2 a q t_2),

cancel = (CANCEL x y e_3 a q t_3), and

deadline = ($t_{current} > t_s$ + timeout)

The initiator x, the participant y, the action to be done a, the relativizing condition q, and the timeout period are formal parameters to this protocol. If the initiator x does not receive any AGREE or REFUSE by the deadline, or if the initiator cancels his request, then the protocol ends (Figure 8.8). Otherwise, if the participant sends an AGREE then either the initiator or the participant may discharge their team commitments by establishing appropriate mutual beliefs.

Protocol to discharge team commitment by the initiator

The initiator either cancels the original request or establishes mutual belief that the action has either been done, or is impossible, or irrelevant, thereby discharging his joint commitment. The precondition for this protocol is that the initiator has a joint commitment with the participant for doing an action and the goal is to discharge that joint commitment. Formally, this protocol can be represented as

$\Pi_{initiator-discharge}$ (x, y, a, q)

= p_1?;((inform-achieved1;g_2?)↑ cancel ↑ inform-impossible1 ↑ inform-irrelevant1);g_1?

where,

p_1 = (JPG x y (DONE y a) (PWAG x y (DONE y a) q)∧q)

g_1 = ¬(JPG x y (DONE y a) (PWAG x y (DONE y a) q)∧q)

g_2 = (MB x y (DONE y a))

inform-achieved1 = (INFORM x y e (DONE y a) t);

(INFORM y x e_1 (BEL y (DONE y a)) t_1)

inform-impossible1 = (INFORM x y e_2 $\Box\neg$(DONE y a) t_2);
(INFORM y x e_3 (BEL y $\Box\neg$(DONE y a)) t_3);

inform-irrelevant1 = (INFORM x y e_4 $\neg q$ t_4);(INFORM y x e_5 (BEL y $\neg q$) t_5)

cancel = (CANCEL x y e_6 a q t_6)

Note that it requires an INFORM and a subsequent confirmation to establish mutual belief about the achievement, impossibility, or irrelevance of the jointly intended action a. However, a single message is sufficient to establish mutual belief that the request has been cancelled, thereby dissolving the JPG. Lemma 2.2 and Lemma 2.3 state when two messages are needed to establish mutual belief and when just one message suffices for this purpose.

Protocol to discharge team commitment by the participant

A participant who agreed to the request discharges his commitment by establishing the appropriate mutual belief. If the participant succeeds in doing the requested action then it will inform that the requested action has been done. If the participant discovers that the action is impossible or irrelevant, then it will establish mutual belief about the corresponding fact. The successful execution of this protocol discharges the participant from the joint commitment it had prior to executing this protocol. This protocol can be represented by the following joint action expression.

$\Pi_{participants-discharge}$(x, y, a, q)

= p_2?;((inform-achieved2;g_2?)$\uparrow$ inform-irrelevant2 $\uparrow$ inform-impossible2);g_3?

where,

p_2 = (JPG x y (DONE y a) (PWAG x y (DONE y a) q)$\wedge q$)

g_2 =(MB x y (DONE y a))

g_3 = $\neg$(JPG x y (DONE y a)) (PWAG x y (DONE y a) q)$\wedge q$)

inform-achieved2 = (INFORM y x e (DONE y a) t);
(INFORM x y e_1 (BEL x (DONE y a)) t_1)

inform-irrelevant2 = (INFORM y x e_2 $\neg q$ t_2);
(INFORM x y e_3 (BEL x $\neg q$) t_3)

inform-impossible2 =(INFORM y x e_4 $\Box\neg$(DONE y a) t_4);
(INFORM x y e_5 (BEL x $\Box\neg$(DONE y a)) t_5)

Here again, confirmation of the INFORM is required to establish mutual belief.

Composing the protocols

The above protocols can be composed to yield a complete request for action protocol. We first compose the last two protocols $\Pi_{initiator-discharge}$ and $\Pi_{participants-discharge}$ to get a combined protocol $\Pi_{team-discharge}$ to discharge team commitments. We repeat the action expressions for the two protocols from above.

$\Pi_{initiator-discharge}$ (x, y, a, q)

$= p_1?;((\text{inform-achieved1};g_2?)\uparrow \text{ cancel} \uparrow \text{inform-impossible1} \uparrow \text{inform-irrelevant1});g_1?$

$\Pi_{participants-discharge}$(x, y, a, q)

$= p_2?;((\text{inform-achieved2};g_2?)\uparrow \text{inform-irrelevant2} \uparrow \text{inform-impossible2});g_3?$

We note that since both these protocols have the same precondition, that is, $p_1 = p_2$, their starting landmarks satisfy the enabling criterion for OR composition. Therefore, $\Pi_{team-discharge}$ defined by the following composition is meaningful.

$\Pi_{team-discharge} = \Pi_{initiator-discharge} \mid \Pi_{participants-discharge}$

Hence,

$\Pi_{team-discharge}$(x, y, a, q) $= p_1?;$[A | B]

where,

A = (inform-achieved1$\uparrow$ cancel $\uparrow$ inform-impossible1 $\uparrow$ inform-irrelevant1);g_1?

B = ((inform-achieved2;g_2?)$\uparrow$ inform-irrelevant2 $\uparrow$ inform-impossible2);g_3?

Next, we want to compose the protocol $\Pi_{form-team}$ to form a team with the protocol $\Pi_{team-discharge}$ to discharge the team commitments. From Section 8.5.3,

$\Pi_{form-team}$(x, y, a, q, timeout)

$= p?;$ request;{($\neg$deadline?;[(agree;g? $\uparrow$ refuse) | cancel]) | deadline?}

Now the precondition p_1 of the protocol to discharge a team is same as the goal g for the protocol to form a team. Therefore, by the "enabling criterion" for embedding a protocol within another protocol (Section 8.5.2), the following composition is meaningful.

$\Pi_{form-team}$[g?/ $\Pi_{team-discharge}$]

That is, we replace the action g? for testing whether the goal g has been achieved, in the action expression for the protocol $\Pi_{form-team}$, by the action expression for the protocol $\Pi_{team-discharge}$.

The resulting protocol $\Pi_{complete}$ is then given by

$\Pi_{complete}$(x, y, a, q, timeout) =

$p?;$ request;{($\neg$deadline?;[(agree;C$\uparrow$ refuse) | cancel]) | deadline?}

where,

C = p_1?;[A | B] is the action expression for protocol $\Pi_{team-discharge}$ derived earlier.

The protocol $\Pi_{complete}$ starts and ends without any joint commitment for doing the requested action between the initiator and the participant. All joint commitments created during the execution of this protocol are discharged when the protocol terminates. Therefore, this protocol is complete and consistent by the composition criteria in Section 8.5.2.

8.6 CONVERSATION PROTOCOLS IN STAPLE

We specify protocols as joint action expressions consisting of communicative as well as other actions. As such, protocols in STAPLE can be viewed as joint plans with some differences in how they are expressed and executed. The joint action expression of a protocol can be executed directly by assuming an overarching joint commitment towards protocol execution [65]. The overarching team commitment can be thought of as a social norm specifying that all STAPLE agents be jointly committed to executing the common protocols provided by the STAPLE protocol library. As such, the STAPLE interpreter executes protocols just like any other joint action expression with some additional support. For instance, the actions within the joint action expression for a protocol must be made system-wide unique. This is achieved by modifying the event identifier of all actions in an instantiated protocol to include a system-wide unique conversation identifier. This identifier scheme also serves to distinguish an action expression for a conversation protocol from any other joint action expression.

Conversation protocols, like other action expressions, do not exist independently on the commitment stacks. The action expression for a protocol gets pushed onto an agent's commitment stack when the agent commits to executing that protocol. This commitment may in turn be relative to the agent's having a joint commitment for executing that protocol. Also, STAPLE agents currently do not distinguish between plans (named action expressions) and protocols during means-ends reasoning. Therefore, if the agent has a plan as well as a protocol that can bring about the same state then the agent will end up committing to an OR expression consisting of the plan and the protocol in order to bring about the desired state. Protocols are annotated as such and a STAPLE agent does not do first principles reasoning when executing communicative actions as part of executing a conversation protocol.

As a result of the overarching policy on conversation protocols, the participants of

a conversation form a temporary team just for the purpose of that conversation. This team has an ad-hoc name created using the list of the conversation participants. One effect of the overarching joint commitment towards protocols is that there is no need to explicitly request to jointly intend a particular protocol. Every message exchanged by STAPLE agents contains information on the protocol being used as well as information on instantiating that protocol. As such the first message of a conversation is taken to indicate the sender's implicit request for executing the protocol in the message (this first message must also be the first communicative action of that protocol). For example, lets consider a protocol π specified as follows.

π = (REQUEST x y a q);
{[(AGREE y x a q);action(y,a);(INFORM y x (DONE y a));
(INFORM x y (BEL x (DONE y a)))] | [REFUSE y x a q] }

This protocol is taken from a more general request for action protocol[4]. An agent x requests another agent y to do an action a. Agent y either agrees or refuses to do the action. Refusal by agent y terminates the protocol by discharging the PWAG of agent x towards agent y created by performing the request. An agree from agent y creates a joint commitment between the two agents that agent y does the action a as discussed in the previous section. Thereafter, agent y does the action a, and establishes mutual belief with agent x about that fact. Also, as discussed earlier, mutual belief is established by an inform followed by a confirmation from the other agent. Lets assume that agent x has a PGOAL that (DONE y a) and it chooses to use this protocol because one of the endpoints[5] of this protocol results in (DONE y a). As a result, agent x has a subgoal (PWAG x y π q) as a means of achieving (PGOAL x (DONE y a)). Here q is the agent's original commitment (PGOAL x (DONE y a)). The (PWAG x y π q) leads to agent x's intending and executing the first action in π that happens to be the REQUEST action. On receiving the request message, agent y may refuse to jointly execute the protocol π by performing (REFUSE y x π q) that discharges agent x's PWAG for executing the protocol π. Alternatively, agent y may either AGREE or REFUSE to do the requested action a. Refusal to do the action a terminates the protocol, and AGREE leads to further actions on part of the agents as specified in the protocol.

[4]This protocol is incomplete, and it does not test for its landmarks ([64]). However, it is good enough for illustrative purposes in this section.

[5]A protocol may have multiple endpoints ([64]) – this is one difference between a plan and a protocol. However, for now, assume the endpoint to be same as the effect in a plan.

One of our defeasible rules in Section 2.4.2 says that performing a communicative action establishes the mutual belief that the communicative action was just done. Therefore, the agents will not attempt to establish mutual belief after performing every communicative action in a protocol that the communicative action was done even when executing the joint action expression for a protocol under lockstep policy. Also, recall that we have rules to establish as a subgoal the commitment implied by the effects of an action that has just been done. Such a rule is needed for reasoning by first principles. However, a protocol designer creates protocols by using the same reasoning process offline. Therefore, the agents executing a protocol should simply use the actions specified in the protocol, and not attempt to find the next action to use by first principles reasoning. This requirement is implemented by modifying the consistency checker, such that an agent will not subgoal a commitment if that commitment is subsumed by an existing higher-level commitment. Another problematic issue in protocol execution is when the protocol has a non-deterministic OR expression of actions with different actors. The default STAPLE rule of leadership-based determination may not be the behavior one expects in this case, and so the agent programmer may need to override the default behavior and specify what to do in this case. Also, the parameters for instantiating the action expression of a protocol are explicitly communicated as part of the messages in order to enable the recipients to instantiate the same protocol properly at their end.

8.7 SUMMARY

The most important aspect of a protocol is its overall purpose, that is, the goal that is achieved by the successful execution of a protocol. Therefore, a protocol should be specified not simply as a pattern of communicative acts but as a sequence of waypoints that must be followed in order to accomplish the goal associated with that protocol. This approach leads to the idea of using partially ordered landmarks for analyzing protocols where landmarks are characterized by the propositions true in them. It was discovered that partially ordered landmarks do not represent just a single protocol, rather they represent a family of protocols, and an agent executing any of these protocols has to go through the same waypoints in order to achieve the goal associated with that protocol. A protocol family for getting tasks done by forming a team was specified, and three different protocols were logically analyzed to show that they all belonged to the same protocol family. The consequences of jointly intending a protocol were discussed and it was noticed

that protocols could be represented as a joint action expression using a rational choice operator along with other operators from dynamic logic. It was also argued that syntactically legal protocol compositions can be specified using the operators from dynamic logic, and semantic criteria were given using landmarks to determine whether or not a protocol composition is complete and meaningful. Finally, the action expression for the Request conversation protocol was derived from the protocol family represented by the landmark expression L1$\succ$ L3 $\succ$ (L4 $\perp$ L5) in Figure 8.9. The following (sub) protocols were specified – a protocol $\Pi_{form-team}$ to create a joint commitment between the initiator and the participant, a protocol $\Pi_{initiator-discharge}$ to discharge the initiator's team commitments, and a protocol $\Pi_{participants-discharge}$ to discharge the participant's team commitment. The protocols $\Pi_{initiator-discharge}$ and $\Pi_{participants-discharge}$ were composed using the OR composition to create a protocol $\Pi_{team-discharge}$ and it was shown that the resulting composition is meaningful using the composition criterion mentioned earlier. Finally, the protocol $\Pi_{team-discharge}$ was composed with the protocol $\Pi_{form-team}$ to get a compact action expression representation of the Request protocol. The resulting composition $\Pi_{complete}$ is again complete and meaningful by the same composition criterion. The actions such as inform-achieved1 were precisely defined earlier. Now the action expression represented by $\Pi_{complete}$ gives a complete representation of the well-known "request for action" protocol [41, 115] in one logical sentence and this protocol is now amenable to analysis using joint intention theory and to execution by joint intention interpreters.

The STAPLE interpreter presented earlier in this dissertation has a built-in protocol library and it can execute fully specified protocols (i.e., those without the rational choice operator) represented as a joint action expression, thereby nearly eliminating the need to implement a separate protocol handling system. All one needs to do is to represent a protocol as a joint action expression and have the agents jointly intend that action expression, say, by specifying (JI x y Π q) in the agent specification files and get the protocol execution for free. The protocol behavior along with the robustness, the mutual belief establishment, and other joint intention specific behavior are obtained automatically subject to the limitations of the implemented interpreter.

To conclude, we have presented a formalism for conversation protocols within the framework of joint intention theory [70, 25]. We regard conversation protocols as having an associated goal that they are meant to achieve and we proposed a formalism for protocol families using partially ordered landmarks that must be accomplished in order to achieve the goal associated a protocol. A landmark is characterized by the propositions that

are true in the state represented by that landmark. We treat conversation protocols as joint action expressions; define composition of protocols using action expression operators; and give criterion for meaningful compositions. We discussed the consequences of jointly intending protocols and argued that one can gainfully apply the joint intention theory to protocols and their compositions. The contributions of this chapter have two main impacts. First, they enable protocol and agent designers to specify and analyze protocols for their correctness, and second, they enable direct execution of conversation protocols just like any other joint action.

Chapter 9

Related Work

The present dissertation spans several research areas in multi-agent systems and there is a large body of literature associated with each of them. We have already discussed some of the related work in the previous chapters wherever it was necessary to provide background for that chapter. For example, the related work on fault-tolerance in multi-agent systems and in traditional distributed systems was discussed in the background for the Adaptive Agent Architecture (Chapter 6). Similarly, the various techniques that have been proposed for representing conversation protocols were discussed as part of background for the chapter on multi-agent conversation (Chapter 8). This chapter points to a comprehensive list of related research publications that have not yet been discussed and it comparatively discusses a representative subset of that list in sufficient detail. We will limit ourselves to the following broad research areas: (1) theories of agency and teamwork, (2) agent development infrastructures, (3) architectures for collaboration and dialogue, (4) agent programming languages, and (5) multi-agent communication and conversation protocols. The first five sections of this chapter discuss related work in each of these five areas and we conclude in the end with a brief summary.

9.1 THEORIES OF AGENCY AND TEAMWORK

The theories of agency and teamwork can be classified into three broad categories – philosophical, logical, and statistical. These theories have been reviewed in great detail in several papers, for example, [120, 121]; theses, for example [117]; and books, for example, [118, 80, 32]. Here, we provide an overview from the point of view of this dissertation and point to the appropriate references for further details.

9.1.1 Philosophical Theories

The notions of intentions, communication, collaboration, and joint actions have been studied in social sciences for many years. Of particular interest is some of the more recent work such as [12, 11, 10, 38, 101, 102, 112] that provide the motivation and criterion for formalizing these concepts in computer science. According to these philosophers, agents have limited resources and therefore, they cannot indefinitely keep weighing their desires, beliefs, and pros and cons of their actions and at some point they must settle down on some choices that they "commit" to doing.

Bratman postulates the following three main properties of intentions that are necessary for resource bounded practical agents and these properties form the basis of most of the theories of agency in computer science. First, intentions lead to means-ends reasoning for agents because they must determine how to achieve their intentions. Second, intentions provide a screen of admissibility for other intentions and an agent cannot adopt an intention if it will make an existing intention impossible. Third, agents must be able to track the success of their intentions and they should be able to retry achieving it if it does not succeed that is, they must not give up their intentions so soon. Also, agents must believe that it is possible to achieve their intentions, and an agent need not intend all the expected side effects of their intentions. It has been shown in [25] that the definition of intention (INTEND) in terms of commitment (PGOAL) in joint intention theory meets these requirements of intention.

Different philosophers have proposed very different notions of collective intention for a group of agents. For example, [112] proposes that an agent must intend to do its part of the group action and believe that it is mutually believed that others will do their part as well. On the other hand [102] proposes that collective intentions cannot be reduced to individual intentions at all and gives examples of situations that tend to support his theory. One of the strong points of the joint intention theory is that it not only explains the examples posed by Searle but is also reducible to appropriate intentions of the agents in a team.

9.1.2 Logical Theories

Several competing logical theories of intention as well as collaboration have been proposed in the literature [91, 104, 45, 17, 90, 57, 119, 39] that aim to model the same artifacts. However, the closest in terms of acceptability and implementations is the Shared Plans [45] formalism for joint action by a group.

Shared Plans Theory

A shared plan for group action specifies beliefs about how to do an action and its sub-actions, and its formal model captures intentions and commitments towards the performance of individual and group actions. This theory distinguishes between (1) intention to do an action, and (2) intention that a proposition hold. An agent intending to do some action must be committed to doing that action, must have appropriate beliefs about its ability to do that action, and it must either have knowledge about how to do that action or how to figure out how to do it. An agent intending that a proposition hold must be committed to doing whatever it can to bring about that proposition (however, it need not necessarily be able to do anything). The components of a collaborative plan include mutual belief of a (partial) recipe, individual *intentions-to* perform the actions, individual *intentions-that* collaborators succeed in their sub-actions, and individual or collaborative plans for sub-actions.

Intentions-that is the crucial aspect of this formalism in that it leads to collaborative behavior. It is used to represent agents' commitments to their group activity, to specify the collaborative support that participants in a Shared Plan offer one another, and it provides a basis for agents to form mutual belief necessary for collaboration. For example, if a participant who is committed to a group activity A whose recipe requires doing action a followed by action b comes to privately believe that a has been done, he will establish mutual belief about this fact if the group mutually believes that mutual belief is required in the group that action a has been done before the group can proceed to action b. However, if the Shared Plan that the group is following does not specify that such a mutual belief is required then the collaboration resulting from that Shared Plan will be fragile in the face of changing private beliefs. Also, note that the concept of intentions-that is an additional concept for capturing commitments whereas in the JI theory, commitment is part of the definition of intention.

The Shared Plans theory distinguishes between a Full Shared Plan (FSP) and a Partial Shared Plan (PSP). A FSP for doing an action a is a complete plan in that all aspects of performing the action a have been completely determined. The group mutually believes that every group member has intention that the group does the action a, the group mutually believes the recipe to use for performing action a, and each step in the recipe has been fully resolved (meaning that group mutually believes who are the actors for that step and what is the full shared plan for that step).

A partial shared plan (PSP) specifies the minimal shared mental state that must exist

for collaboration to exist. The various aspects of the shared plan (such as recipe, actors, etc.) can be partial but the group must mutually believe that (i) every group member has commitment (intention-that) for doing the group activity, (ii) every group member is committed (i.e., has intention-that) to identify the parameters necessary for doing the group activity (such as missing steps in a recipe, the actors of various steps, etc.), and (iii) every group member is committed (i.e., has intention that) to making sure that the constraints for the group action will hold. Similar to PSP, recipes can be partial or complete, and individual plans can be partial or complete. In case of PSP, the Shared Plans formalism requires the group to mutually believe and be committed to the process (i.e., be committed to a full shared plan) that the group will use for evolving a PSP into a FSP (for example, how to assign subagents, how to select a recipe, etc.).

The shared plans formalism is recursive in nature in that each of the steps in a recipe for a shared plan must have an associate shared plan for doing that step. This recursive definition leads to collaboration between subgroups of agents for doing parts of a higher level shared plan. In JI theory, there is no explicit recursive joint intention between subteams for doing parts of the jointly committed action - it is left for the team to choose the best possible course for doing a sub-action.

Team Formation

Joint intention theory provides a mechanism for team formation (for example, using Request-Accept pair of communicative actions) and prescribes the behavior of a team once it is formed. However, it is silent about the process of finding teammates, and when agents should come together to form a team. The cooperative problem solving process proposed by [119] attempts to address this issue. It proposes the following four stage process: (1) recognizing the potential for cooperative action, (2) attempting the team formation, (3) plan formation, and (4) team activity. These processes are formalized using constructs similar to that of joint intention theory but the main differences are in the first step (recognizing the need for team) and the third step (explicit plan formation by the team).

BDI Model

The Belief, Desire, Intention (BDI) model of Rao and Georgeff [90] is closest to joint intention theory as far as the modeling of individual agents is concerned. JI theory has beliefs and goals as the primary modal operators and it defines intention using these operators

whereas the BDI model adds intention as a primary modal operator at the same level as beliefs and goals, and postulates axioms that relate intention to the other modalities. The possible worlds semantics of joint intention theory uses linear time temporal logic whereas the BDI theory uses a variant of branching time temporal logic called CTL*. These differences in the underlying logical systems result in different properties of the constructs in these two theories. The theoretical differences apart, the teamwork extension [91] of the BDI model has neither been widely accepted nor has it been implemented in software systems. However, an agent development platform called PRS (Procedural Reasoning System) based on this model has been one of the most popular ways to develop BDI agents (meaning agent programs that use explicit representation of Beliefs, Desires, and Intentions). PRS is discussed again in Section 9.2.1.

9.1.3 Real Time and Statistical Theories

Ortiz and colleagues [81, 122] have proposed techniques for real-time coalition formation and cooperation between hundreds of mobile semi-intelligent, resource bounded sensors. The real time issues addressed in this research are very useful but they are not the focus of the present dissertation.

Tambe and colleagues [75, 87] have demonstrated the usefulness of POMDP (Partially Observable Markov Decision Process) based models in statistical reasoning about communication, and in comparing the communication policies of various implementations inspired by joint intention theory. This body of work can be construed to be complementary to the STAPLE research. For instance, one can imagine using decision-theoretic analysis in STAPLE for reasoning about communication costs in order to make decisions about issues such as how to best establish a mutual belief, when to delay the establishment of mutual belief, and how to select the best possible action from a non-deterministic OR expression of candidate actions to achieve a particular goal.

Next, we look at some agent infrastructures used for developing single agents (as opposed to collaborating and communicating multi-agent systems).

9.2 SINGLE AGENT INFRASTRUCTURES

There is a long list of agent development platforms and single agent architectures that offer different capabilities. Some of them are BDI systems in that they have explicit representation of beliefs, desires, and intentions whereas others simply provide a flexible way

of quickly putting together an autonomous agent. The most well known systems in this list include the PRS from SRI, RETSINA from CMU, JATLite from Stanford, JACK (a commercial agent development platform), and implementations of the FIPA agent platform (a standard for agent development and communication). Most infrastructures in this category can be used for developing multi-agent systems that communicate using a standard communication language such as FIPA and KQML. However, support for teamwork and communication is not built into these systems but these are extra functionality added by the agent programmer.

9.2.1 The Procedural Reasoning Systems (PRS)

The procedural reasoning system [43] developed at SRI played an important role in popularizing BDI agent architectures. Over the years, several competing implementations based on PRS became available that attempt to either enhance PRS or to fix some of its drawbacks. We collectively refer to the original PRS as well as its variants as the PRS systems.

STAPLE owes its planning terminology and constructs, action execution methodology, abstractions for sensors and effectors, and stack-based tracking of commitments and intentions to the PRS systems. Specifically, it inherits from and significantly builds upon the experiences from the PRS system called JAM [51]. Although the body of a STAPLE plan is represented as a complex action expression that can be reasoned about by a joint intention interpreter, it is conceivable to elaborate the plan representation language (for example, by introducing a notion of cue or hint) to be more in line with recent PRS systems such as SPARK [74]. Actions in STAPLE are similar to procedures – they can be defined in any language (currently only Java, and Prolog) but they must have at least an effects part represented using the logical language used by STAPLE (actions may have other optional components such as precondition). It enables the STAPLE interpreter to abstract away from the otherwise procedural construct and reason about them logically. One contribution of STAPLE to the PRS legacy is its close connection with a logical theory of agency. Whereas the BDI logic provides the specification for a PRS implementation, the agents in STAPLE are specified using the logical language of joint intentions and the STAPLE interpreter attempts to faithfully execute the specification as per the theory.

A significant difference between STAPLE and PRS systems is the support for teamwork and multi-agent communication. PRS systems such as JAM and SPARK are intrinsically single agent systems with no built-in support for teamwork and in which communication

is usually an add-on feature with little connection to the underlying BDI theory. On the other hand, the notion of teamwork and multi-agent conversation is at the core of the underlying theory behind STAPLE such that STAPLE agents reason about their joint intentions as well as about the ongoing conversations. Unlike KQML and FIPA, the communication in STAPLE is based on a provably correct formal semantics of multi-agent conversations presented in this dissertation.

9.2.2 Intelligent Resource-bounded Machine Architecture (IRMA)

The Intelligent Resource-bounded Machine Architecture [12] is similar to PRS in that it has a plan library, and explicit representations of beliefs, desires, and intentions. Its infrastructure has a reasoner for reasoning about the world, a means-ends analyzer for determining plans to achieve the agent's intentions, and an opportunity analyzer for monitoring the environment and determining further options for the agent. It also has a filtering process for determining the subset of the agent's potential courses of action that are consistent with the agent's current intentions. A deliberation process makes the choice between competing options. The various components of IRMA are already present in the STAPLE implementation presented in this dissertation. IRMA supports execution of only action at a time by the agent (as opposed to simultaneous execution of multiple actions in STAPLE), and it does not have support for multi-agent conversations and teamwork.

Next, we look at some multi-agent infrastructures that support teamwork and communication.

9.3 AGENT INFRASTRUCTURES FOR TEAMWORK AND COMMUNICATION

As far as teamwork is concerned, STAPLE is related to STEAM [110] and GRATE* [52] both of which are agent architectures with teamwork capabilities. As far as collaborative dialogue is concerned, STAPLE is related to ARTEMIS [14] and Collagen [93]. STAPLE is also related to Breazeal's infrastructure for human-robot collaboration that builds upon both joint intention and shared plans theory similar to that done by STEAM. Breazeal's system forms the motivation for the "Lights World" domain used in the present dissertation and it was discussed in Chapter 3.

9.3.1 STEAM and TEAMCORE

The teamwork support in STAPLE is based on the same joint intention theory that inspired STEAM [109]. STEAM also addresses some other important aspects of teamwork such as decision theoretic reasoning about communication that is complementary to the logical reasoning performed by STAPLE agents.

STEAM has team operators, it supports rules, and is inspired by joint intention theory. It also has explicit representation of team goals, plans and joint commitments. It introduces a recursive method of forming sub-teams for doing parts of a group activity. A new joint commitment is created between the sub-team for doing the sub-action. This results in an explicit hierarchical joint intention similar to that in the Shared Plans technique. Jointly intended actions are allowed to be partial and sub-teams are created to resolve partially specified actions.

STEAM implements the joint commitment and other teamwork concepts as reusable rules and the same STEAM codebase has been used for a many collaborative applications. One difference between our work and STEAM is the language for teamwork specification. STAPLE uses a logical language and can interpret complex logical statements of joint intentions in contrast with STEAM, which does not use an explicit action interpretation language. As such, the current work has the advantage of allowing one to modify team behavior at a very high level by modifying a sentence in the logical specification of agents, as well as the potential benefit of being able to logically prove the behavior of a system from its specification. Importantly, this logical language can be the target representation for a multimodal interface, thereby enabling a user to speak and draw (or use some other interface modalities) in order to instruct a team. The focus of our current research is on translating logical specifications of teamwork and a formal semantics of communication into executable agents. On the other hand, STEAM attempts to incorporate joint intentions on top of a very different cognitive agent architecture based on the SOAR model [109], and STEAM agents do not logically reason about communication. As such, the coupling between STEAM and joint intention theory is indirect. STEAM includes a comprehensive decision theoretic analysis of communication among team members that can be used to give agents a flexibility to choose between different communication models. As compared to STEAM, the main contribution of STAPLE to the teamwork research is in taking the STEAM concept of directly executing team specifications to the next step by offering interpretation of agent specifications in a logic that is used for formal specification of joint intention theory.

STEAM is the infrastructure behind TEAMCORE that has been used for a number of successful team-based applications. TEAMCORE provides proxy agents that wrap legacy and non-team aware systems to make them behave as a team-member to the rest of the agent team. It is a very interesting and useful concept but there are no formal guarantees within TEAMCORE that the proxy agent does in fact makes a non-team aware agent appear as a team member. On the other hand, STAPLE can be used to build upon this TEAMCORE concept by providing the ability to prove that a non-team aware agent does in fact behave as a team member. STAPLE offers the advantages of dynamic program synthesis – legacy code accompanied by its description in the formal language can be treated as an "action" by an agent. The agent then reasons about those actions using their formal description, decides upon and executes the best course of action(s) at any given time, and even switches between intentions (and therefore, between the legacy codes being executed) as the priorities of the agent change. One can argue that this approach not only lets one "agentify" any system but also prove that the modified system does in fact have the required agent-like properties. In particular, it should be possible to treat a non-team aware agent as part of a team by using a proxy agent written in STAPLE that reasons using the formal representation of actions of the non-team aware agent, and be able to prove that the non-team aware agent does in fact behave as a team member (and establish conditions under which the proxy agent fails to behave as a team member).

9.3.2 GRATE*

The work by Jennings [52] on the GRATE* agent architecture demonstrated the usefulness of the joint intention theory in real world applications for the first time. It showed that as the world gets more complex and unpredictable, the teams as a whole waste much fewer resources than a group of uncoordinated or self-interested agents. This research distinguished between joint commitment to achieve a goal p, that is, (JPG x y p q), and the joint commitment to do the actions a to achieve p, that is, (JPG x y (DONE a) p). The combination of such joint commitment pairs was then called joint responsibility. However, from a logical viewpoint, this distinction is immaterial, as both joint commitments will have identical behavior, per the definition of JPG. By faithfully interpreting joint intentions, STAPLE implements what was termed joint responsibility in GRATE* without needing any new constructs.

9.3.3 Collagen

Collagen [93] implements Grosz and Sidner's collaborative discourse theory. This theory has three components - linguistic, attentional, and intentional structures. The linguistic structure is implemented via discourse segments purpose, the attentional structure is implemented using focus stacks, and the intentional structure is implemented as a simplified form of Shared Plans formalism for capturing beliefs and intentions. These intentions are used on the focus stacks for tracking the progress of a discourse.

A collaboration manager keeps track of the linguistic and attentional state of discourse, as well as the collaborative intentions of the participants. It recognizes intentions of utterances and actions and tries to fit them into the current intentional state while also allowing for digressions and sub-intentions. To successfully complete collaboration, two agents must mutually believe that (a) they have a common goal, (b) they have agreed on a plan or recipe, (c) they are each capable of their part, (d) they intend to do their actions, and (e) they are committed to the overall success of the collaboration, not just each of the parts. The objective is not about doing physical actions but about the attainment of knowledge necessary to perform actions.

Theoretical differences apart, STAPLE is a more declarative general purpose agent programming language in that it supports first principles reasoning about physical actions as well as communicative acts, whereas Collagen gets its dialogue behavior by implementing the algorithms and discourse structures of the Shared Plans theory [46] and it does not do first principles reasoning about communication or teamwork.

9.3.4 ARTEMIS

This work by Bretier and Sadek [14] resembles the present research in as far as reasoning about communication is concerned. ARTEMIS is based on a single agent variation of the joint intention theory and therefore, the implementation reasons only about individual commitments and intentions. STAPLE is based directly on joint intention that better models the notion of dialogue as a joint activity, and reasons not only about internal commitments and intentions but also about social commitments, and mutual beliefs of agents. ARTEMIS defined communicative acts such as INFORM and logically reasoned about communication in a similar fashion to that done by STAPLE. We will revisit the communication language used by ARTEMIS (called ARCOL) in the next section.

Next, we look at some declarative agent programming languages similar to STAPLE.

9.4 AGENT PROGRAMMING LANGUAGES

STAPLE belongs to the same family of agent programming languages as Agent0 [103], AgentSpeak(L) [89], Concurrent Metatem [42], ConGolog [44], 3APL [49], and IMPACT [108]. The main distinguishing characteristic of STAPLE is the support for teamwork and formal communication semantics that are directly connected with a logical theory of agency. Most of the other aforementioned languages are essentially single agent programming languages even though they may support communication to some extent using agent communication languages such as KQML in an ad-hoc fashion. Some of these languages provide the important first steps towards bridging the gap between agent theory and practice in a comprehensive manner. However, a unified formal framework for theory of agency, communication, and teamwork in an agent programming language is still missing, and the current research is the beginning of an effort towards addressing this gap.

The language Agent0 uses modal constructs like belief and intention and has syntactic support for these concepts but the formal link between the language and the modal logic is not clear. AgentSpeak(L) is a rule-based language that provides an alternative formalization of BDI agents in terms of operational and proof-theoretic semantics, and can be viewed as an abstraction of BDI systems such as the PRS [43]. The beliefs of agents are represented explicitly but intention is regarded as a stack of partially instantiated plans. Compared to AgentSpeak(L), STAPLE provides a richer set of constructs, and a more expressive action theory, along with reasoning about teamwork and communication. In Concurrent Metatem, a set of rules in temporal logic is used to describe the behavior of an agent and this logical specification is executed directly to obtain the desired agent behavior. The direct execution of a formula is an attempt to build a model for that formula using iterative model generation techniques. Unlike STAPLE, the agent modalities such as goal in Concurrent Metatem are not independent declarative concepts but are dynamic logic formulas (for instance, eventualities such as $\Diamond$p can be taken to represent an agent's goal). An interesting feature of Concurrent Metatem is the support for groups. Groups can be either named groups, or can be a set construction formula where any agent satisfying the formula is considered as a member of the group. However, unlike in STAPLE, groups are not part of the logic, and there is no formal group communication semantics.

ConGolog [44] is also a direct execution logical language based on the situation calculus. It has a richer set of action expression constructs than STAPLE, and it supports imperative programming (via procedures). Agents are programmed by specifying a set of axioms such as those describing the initial state, the precondition for each primitive

action, and the successor state axioms. Agent concepts such as beliefs and goals are not independent concepts but are modeled within the situation calculus. For instance, the situations of the situation calculus can be viewed as belief states of an agent, and a high-level program can be considered to be the goal of an agent. Multi-agent interaction is supported via abstract communicative acts defined as procedures but communication and coordination (such as teamwork) is not part of the basic logic. The operational semantics of ConGolog programs is specified using Plotkin-style transition systems within the formalism of situation calculus. The agent programming language 3APL [49] is similar to ConGolog in that it combines features of logic programming (proof as computation) and imperative programming (procedural constructs). An agent in 3APL consists of a goal base, a belief base, and a practical rules base. The operational semantics of the language is specified using transition systems, and a variant of modal logic is used to specify its denotational semantics. 3APL is not a direct execution system like STAPLE, and teamwork and groups are not part of the language or the logic.

The IMPACT framework [108] is a comprehensive agent platform, much like the FIPA framework, with support ranging from yellow pages, and registration, to thesaurus, and agent launchers. The semantics of IMPACT agents is based on deontic logic, and agent programs consist of rules and actions. The rules may contain deontic modalities, and actions are specified using executable legacy code and a logical code-call condition. As such, the action specification in IMPACT is similar to that in STAPLE and allows agentification of arbitrary code. Teamwork and agent communication are not formally integrated into the logical language behind IMPACT.

Thus, the main contribution of STAPLE lies in enabling direct execution of agent specifications in a unified logical framework of teamwork, multi-agent communication, and theory of agency. Next, we look at some research related to agent communication languages and conversation protocols presented in Chapter 2, Chapter 5, and Chapter 8.

9.5 AGENT COMMUNICATION LANGUAGES AND CONVERSATION PROTOCOLS

There are three issues related to multi-agent communication that concern us in the present dissertation – semantics of communicative acts, semantics of group communication, and semantics of conversation protocols.

9.5.1 Semantics of Communicative Acts

There has been substantial research on formal semantics of agent communication languages over the past decade resulting in very rich and varied agent communication languages. The two widely used agent communication languages are FIPA [41] and KQML [67], each of which provides some form of formal semantics. These formal semantics provide scaffolding for interoperability in that they convey to agent designers exactly the same meaning and intended usage of a communicative act. As such, the role of formal semantics in most implemented systems has been relegated to helping agent designers decide what messages are to be exchanged during inter-agent conversation. However, one of the main benefits of such formal semantics is the ability to reason about communication which, in principle, should automatically result in dialogue behavior. In fact, ARCOL [14], the precursor of FIPA, was the core of a spoken dialogue system called ARTIMIS [14]. This system reasoned about agent communication using logical definitions that eventually found their way into the formal semantics of the FIPA communicative acts. However, a drawback of the formal semantics specified by FIPA is that it does not provide motivation for when to communicate, what to communicate, how to handle failure, or how to conduct dialogue (beyond fixed protocols). For example, suppose that agent x asks agent y to do something to which agent y agrees. However, before y could do that task, agent x decides that it no longer wants that task done. There is nothing in the FIPA semantics that forces agent x to inform this fact to agent y, meaning that agent y may end up unnecessarily wasting resources. So even if agents reason using the semantics specified by FIPA, they are still subject to these limitations. On the other hand, a semantics of communicative acts based on a theory of teamwork (such as joint intention theory [70]) will require agent x to inform agent y that it has changed its mind.

9.5.2 Semantics of Group Communication

The major agent communication languages have either no provision or no well-defined semantics for group communication. For instance, in the FIPA ACL, the only way to inform a set of agents is to inform them individually, one at a time. Furthermore, semantics of the FIPA communicative acts imposes the precondition that the sender has certain beliefs about the mental state of the (known) addressee. Consequently, there is no way to send messages to unknown agents – a typical scenario in broadcast communication. KQML does offer several primitives, such as broadcast and recruit-all, that have group flavor but these primitives are merely shorthand for a request to do a series of other

communicative acts. Proper semantics cannot be given to group requests such as "One of you, please, get me a slice of that pie." We can conclude that the semantics of group communication presented in Chapter 5 is still missing from the agent communication languages though some researchers have started to implement systems [16, 15, 95] based on the semantics presented in this dissertation and to use group communication in areas such as argumentation [79]. The research on group communication presented in this dissertation has already been applied towards group protocols [96].

9.5.3 Semantics of Conversation Protocols

Even though researchers in natural languages have been claiming that dialogue is a joint action for sometime, the multi-agent systems community has started to explore this issue only recently. Vongkasem and Chaib-draa have argued that a conversation in the context of an agent communication language is a joint activity that can be realized as sequences of smaller actions [113]. They informally identify the various aspects of this approach and propose to view conversation protocols in terms of joint commitment. The informal ideas presented by Vonkasem and Chaib-draa is to some extent similar to our formal analysis of joint commitment towards protocol families represented as landmark expressions and joint intention towards concrete protocols represented as action expressions (Chapter 8).

A key idea in this dissertation is that protocols are meant to perform certain tasks, that is, successful execution of a protocol achieves the goal associated with that protocol. Some research that has touched upon this aspect of protocols can be found in [107, 40, 84], but none of this prior work presents a complete or integrated approach. Work by Pitt and others explicitly links "successful outcomes" (ostensibly, goals) to conversation protocols but does so by annotating a syntactic framework with semantic summary expressions in such a manner that the two are not directly connected (and, as such, may become semantically incongruous if anything in either the syntactic protocol or the semantic summary are modified in isolation). Elio and Haddadi [40] discuss dialogues for joint tasks and about jointly maintaining global coherence in a conversation. They informally explore their insights that protocols are task oriented but have not provided a concrete formulation of their research. Dignum and colleagues have analyzed the process of team formation using structured dialogue in a modal logic [39]. We use a different modal logic than these researchers but the basic idea of team-formation using conversations is not a new one. This idea was formally demonstrated in [106, 107] wherein it was shown that joint commitments are created using a sequence of speech acts.

The research community has not given much attention to the formal semantic ties between individual communicative acts and patterns of such acts. The present dissertation extends the work started by Smith and colleagues [106, 107] towards formally integrating protocols and individual communicative acts. Pitt and Mamdani [86] and Yolum and Singh [123] have subsequently addressed this issue using a different approach than ours. Most researchers, including us, regard communicative acts as the basic concept and view protocols as a pattern of these acts. Our approach to analyzing correctness and completeness of protocols using the semantics of communicative acts reflects this relationship between protocols and communicative acts.

Pitt and Mamdani [86, 85] differ from most researchers in taking conversation protocols as the starting point for inter-agent communication, in which the semantics of single utterances are defined within the context of a syntactic protocol definition (with some semantic attachments). Specifically, protocols are defined using performatives that are themselves given an 'action-level' semantics using the protocol that they are used to define. The action-level semantics of a performative used in a protocol is simply an *intention to reply* using a performative allowed by that protocol.

Singh [105] and subsequently Yolum [123] argue that finite state protocols for agent communication are not sufficiently expressive and lack proper semantics. They provide a formalism using finite state machines in which the states and actions are defined using social commitments among the agents. Singh and Yolum associate a meaning with each state by specifying which commitments are in force in that particular state, and give a meaning to each action by defining how the commitments are affected by that action, thereby leading to a state transition. The basic ideas in this approach are similar to those described in this dissertation and in our previous work [106, 107], with the difference that we use 'joint commitments' rather than 'social commitments'. We believe that the social and mental commitments are related and it might be possible to express one in terms of the other. A PWAG relativized to another agent's desires defines a commitment of one agent towards another using a mentalist notion. Therefore, it represents a social commitment provided that it is made public. PWAG is used in the definition of communicative acts such as the REQUEST. An agent's performing a communicative act makes its intentions public, and therefore performing a communicative act involving PWAG leads to a social commitment. Furthermore, the reasoning rules for 'meta-commitment' used by Singh and Yolum are similar to discharging joint and individual commitments in our framework. In both approaches, one can logically prove that a sequence of communicative actions does in

fact achieve the specified states (or landmarks). The most pronounced difference between the two approaches is that Yolum and Singh do not integrate independently motivated speech acts into their framework.

9.6 SUMMARY

This dissertation spans a large number of research areas in multi-agent systems and correspondingly there is a huge list of related literature. We presented comparative overview of some of the main related research in agent theories, agent architectures, agent programming languages, and multi-agent communication. An interesting finding during the survey of related is that researchers have started building upon the ideas presented in this dissertation for their own research.

Chapter 10

Concluding Remarks and Future Work

The previous chapters presented the problems addressed in this dissertation and its contributions in detail. Here we briefly review the main points of this dissertation and discuss some of the future research.

10.1 OVERVIEW

This dissertation investigated the feasibility of building a logic-based declarative agent programming framework that enables one to declaratively specify joint action resulting in team behavior along with correct task and team-oriented communication. We hypothesized that it may be possible to create such a framework by appropriately extending the theories of teamwork and communication and by borrowing from research on planning and programming languages. This framework would have teamwork and communication primitives whose semantics conform to that predicated by the underlying theory. We also saw that several agent programming languages have been proposed in recent years that attempt to bridge the gap between logical theory of agency and agent implementations. Most of these languages focus on individual agents and their support for agent teams and multi-agent communication is either non-existent or is at best ad-hoc. On the other hand, joint intention theory has successfully been used for rule-based creation of teams, and to provide a formal semantics for communicative acts but, prior to this dissertation, there were no agent programming languages linked to this theory. In order to validate our hypothesis, we presented an agent programming language called STAPLE (Social and Team Agents Programming Language) based on Joint Intention Theory and showed that it supports teamwork and communication in a unified framework.

We used a three phase approach in our investigation – theory, implementation, and validation. In the theoretical phase, we specified a formal semantics of speech acts and

multi-agent conversation showing how they can be used to create and discharge teams. This phase included enhancements to the JI theory itself to support groups of agents and a wider variety of teams such as persistent and dynamic teams. (2) Thereafter, we presented STAPLE using concrete examples of STAPLE agent programs. We discussed an implemented interpreter for the single agent constructs of this language and then modified it to handle commitments of one agent towards another. The operational semantics of this language implemented by the STAPLE interpreter is derived from its logical semantics (JI theory) presented in this dissertation and is discussed in the appendix. (3) Finally, we argued that teams of agents written in STAPLE coordinate and communicate as per the theory, without our having to program these behaviors explicitly. We verified this claim by using the above interpreter to execute agents written in this language for two different domains.

The remainder of this chapter is organized as follows. We recapitulate the need for the present research in the next section. Section 2 summarizes STAPLE and its relationship to logic. The main accomplishments of this dissertation are summarized in Section 3 and possible directions for future research are discussed in Section 4. Finally, we conclude in Section 5 with a brief summary.

10.2 RECAP OF MOTIVATIONS FOR THE PRESENT RESEARCH

The main motivation behind the present research is to build systems that exhibit collaborative teamwork. However, instead of our explicitly programming team behavior, collaboration should automatically follow in virtue of our embedding the foundational concepts of joint action/intention into a belief-desire-intention (BDI) architecture that is equipped with a repertoire of communicative actions. Thus an agent built around this architecture should engage in dialogues and exhibit characteristics of teams such as robustness as a consequence of reasoning about joint action, intention, and belief.

One popular approach to the building of intelligent agents is via the Belief-Desire-Intention (BDI) architecture [90]. This architecture models an agent's acting based on its intentions and commitments, subject to its beliefs. Numerous implementations of this model have been developed, including SPARK [74], the Procedural Reasoning System (PRS), etc. Although these models have often pointed out the need for actions that model communication, prior to this dissertation, little research into the classical BDI architecture

had been directed at the integration of communication, collaboration, and action.

On the other hand, previous research guided by the plan-based theory of communication [1, 33, 29] has shown that communication can be generated through the planning and execution of speech acts. This approach has motivated research both on dialogue systems [14] and on inter-agent communication languages, such as FIPA [41] and KQML [67]. In particular, ARCOL demonstrated that a rational agent implemented as a modal logic theorem-prover could, in fact, participate in dialogues, including spoken language dialogues conducted over the telephone network. However, a major drawback of the formal semantics specified for FIPA and KQML is the lack of motivation for when to communicate, what to communicate, how to handle failure, or more generally, how to conduct dialogue (beyond fixed protocols). For example, suppose that agent x requests agent y to do an action to which agent y agrees. However, before y can do that task, agent x decides that it no longer wants it done. There is nothing in the plan-based theories of communication, or in the semantics underlying FIPA, that causes agent x to inform this fact to agent y.

In recent years, the BDI approach has been extended to model agents who need to collaborate and communicate with others via concepts of shared plans [46] and joint intentions (JI) [70]. JI theory specifies agents' mental states as they execute joint actions and it has been integrated with models of communicative acts to build multi-agent systems [52, 109] that generate collaborative task-oriented behavior. In the example above, JI theory would require agent x to inform agent y that it has changed its mind. This dissertation shows that a JI interpreter should be able to serve as the basis for a collaborative dialog engine. Furthermore, the aforementioned systems have demonstrated the robustness and efficiency of teams of agents as a result of the collaborative behavior prescribed by the JI theory. This dissertation shows that interesting and useful team behavior such as various kinds of fault-tolerance follow when the declarative specification of that behavior is interpreted by a JI interpreter. The JI interpreter implemented as part of this dissertation is summarized next.

10.3 SUMMARY OF STAPLE AND ITS RELATIONSHIP TO LOGIC

The STAPLE language consists of constructs from the JI theory and the STAPLE interpreter attempts to implement the semantics of these constructs as described in this dissertation. Programming constructs (such as details of action specification) are borrowed from the prior work on BDI architectures for aspects of a concrete agent specification that are not part of the theory. We first review the implementation of a STAPLE JI interpreter along with its support for communicative actions and belief reasoning and then discuss its relationship to the JI logic.

10.3.1 The STAPLE Interpreter

The STAPLE interpreter is a multi-stack BDI interpreter with built-in support for PGOAL (commitment), INTEND (intention), and PWAG (directed social commitment). Using these foundational concepts, this interpreter can model joint commitment and joint intention. During the execution of an individual or joint action, an *Intention-Commitment (IC) stack* is built in a bottom-up manner, with the bottom element containing the original commitment and the other commitments or subgoal used to achieve the original commitment layered above it. When a PGOAL is being adopted or subgoaled, the interpreter first checks for its achievement, impossibility, and irrelevance by executing the belief reasoner in a Prolog engine over the agent's belief base. For instance, to check for impossibility, the belief reasoner is invoked to check if $\Box\neg p$ can be concluded from the belief base (this reasoning will make use of any domain dependent applicable rules of the form $\Box\neg p$:- q). Thereafter, a consistency checker that employs the belief reasoner is invoked to make sure that the content of the PGOAL is consistent with the existing commitments and intentions of this agent, and triggers on the belief base are created to monitor the escape conditions in the definition of PGOAL, and in the case of PWAGs, mutual belief of those conditions. The appropriate trigger is fired when one of the above three cases becomes true. The execution cycle is completed when one of the triggers for the original commitment is fired. During the interpretation of INTEND, primitive actions are executed directly, while complex actions are decomposed. It is possible to have multiple stacks because an agent can have more than one commitment at a given time, such as when it is performing concurrent tasks.

An agent specification in STAPLE consists of an initial mental state (beliefs, goals,

commitments, and intentions), capabilities (actions and plans that the agent is capable of performing), inference rules, initial state of the world, domain specific inference rules, etc. Agents in STAPLE are programmed using the usual Prolog syntax extended by operators for dynamic logic of actions (concurrent actions, test, repetition, etc.), temporal logic (eventually and always), and for negation, implication, conjunction, and some other miscellaneous constructs. Primitive actions may optionally specify a precondition that must be true before the action can be executed and a list of desired effects that may eventually become true as a result of the action being performed but there is no guarantee that they will ever become true. These conditions are used in a standard backward-chaining reasoner. Plans in STAPLE are expressed as complex action expressions constructed using the action formation operators for sequence, non-deterministic OR, concurrent action, test action, and repetition. The interpreter recursively decomposes such expressions, executing the self's actions, interpreting the action combinators, and waiting for the execution of other agents' actions to which it has committed. All primitive actions in a plan must have already been specified as mentioned above. Further, since plans are also actions (though not singleton actions), they are required to specify the effects that must be true after a successful execution of the action expression for that plan.

Joint intention theory leaves room for the agents to decide upon the best course of action consistent with the theory. For example, one of the declarative rules in STAPLE specifies that if an agent is committed to achieving a proposition p and it knows of actions that it can do to achieve that proposition, then the agent will intend to perform a non-deterministic OR expression of those actions with respect to the original commitment. We leave open precisely which of these actions to choose, for example, by maximizing utility. Similarly, there are several other rules defined in the rule base that characterizes rational behavior in different situations.

10.3.2 Integrating Communicative Actions

The semantics of communicative acts based on joint intention theory characterizes them as attempts having an associated goal and intention. The goal associated with performance of a communicative act (i.e., its desired effect) is what the agent would like to bring about by performing that act. The intention associated with attempting a communicative act (i.e., its intended effect) is what the agent is committed to bringing about via performance of that act. The STAPLE constructs for defining actions and plans are used to define the communicative actions and these actions are similar to any other action in

STAPLE. The semantics of communicative actions are used to specify the various parts of an action definition, for example, the intention part in the semantics of a communicative act specifies the effect of that communicative action when specified in STAPLE. Similarly, the precondition of a communicative action in STAPLE is derived from and is consistent with the semantics of the communicative act.

We defined two primitive communicative acts, REQUEST and INFORM. The goal of a request (REQUEST x y e a q t) is that the requestee y eventually does the action a and also come to have a PWAG with respect to the requester x to do a. The requester's and requestee's PWAG's are relative to some higher-level goal q. The goal of an inform (INFORM x y e p t) is that the listening agent y comes to believe that there is mutual belief between him and the informing agent x that the proposition p is true.

For the communicative act INFORM to be performed, the informing agent x has a precondition that it believes the proposition p and x does not believe that informee agent y believes p. These operator definitions are used in the usual backward-chaining fashion. Other communicative acts such as CONFIRM, INFORM-REF, INFORM-IF, ASK_REF, ASK-IF, PROPOSE, AGREE, REFUSE are composed using the basic communicative acts INFORM and REQUEST. Composed communicative actions are defined in STAPLE as plans (i.e., as named action expressions). This dissertation showed that a REQUEST from x to y followed by an AGREE from y to x establishes a JPG between the requester and the requestee to do the requested action. Similarly, we showed that it requires an INFORM from x to y followed by another INFORM from y to x that the informed proposition p was believed in order to establish mutual belief about p. Results such as these are used as reasoning rules by the STAPLE interpreter.

10.3.3 Integrating Belief Reasoner

The STAPLE interpreter includes a Horn-clause belief reasoner that implements weak S5 semantics and is capable of reasoning with quantified beliefs within the Horn subset of first-order logic. The beliefs of a STAPLE agent, including those beliefs common to all STAPLE agents, are stored in a knowledge base that supports concurrent access. The belief reasoner is used in the STAPLE interpreter for querying the belief base instead of deducing all possible consequences that can be inferred from the belief base. The system properly reasons with disjunctive and "quantified-in" beliefs [1].

A belief base maintenance system complements the belief reasoner and is primarily needed to help it avoid circular loops and infinite recursions For example, (BEL x (BEL

x... (BEL x p)...))) is reduced to the equivalent fact (BEL x p). Beliefs about other agents are represented just like any other fact in the belief base. For instance, "I believe that x believes p" are asserted into the agent's belief base as (BEL x p), which is a simplified form of (BEL self (BEL x p)). The consistency checker uses the belief reasoner to attempt to ensure that an agent does not adopt any commitment or intention that conflicts with any existing commitment or intention of that agent. For instance, an agent cannot adopt an intention or commitment if it believes that the new commitment or intention makes an already existing commitment or intention impossible. There are other similar rules for maintaining consistency in the system.

10.3.4 Relationship to logic

It is apparent from the above description that we chose to implement the semantics of constructs from JI theory in a different way as compared to traditional theorem proving techniques. The STAPLE interpreter attempts to satisfy a core modal formula such as PGOAL using its definition in the JI theory. Thus, this direct execution of specifications can be viewed as implicit construction of a model for the logical formulae. We treat the model theory of the logic as specifying the denotational semantics of the logical language, specify an operational semantics of the language based on the denotational semantics, and use the operational semantics for the actual implementation.

The semantics of the modal concepts PGOAL, INTEND, and PWAG is built into the interpreter by using the logical definition of these terms. For instance, the definition of PGOAL in the logic says that an agent having a PGOAL that p relative to a higher level goal q does not believe that p and has a goal that eventually p at least until it believes that p has been achieved or is impossible to achieve or is irrelevant. This prescription from the definition is explicated in the operational semantics of PGOAL (see appendix) in a way that can be directly implemented.

Action expressions are not independent terms – rather, they are to be used within the context of a modal operator. As such, the interpretation of action expressions within the context of these terms is implemented by encoding the results of intending (or committing) an action expression as part of the PGOAL and INTEND interpreters.

The belief reasoner is the only STAPLE component that uses theorem proving. It encodes the belief axioms used in the JI theory using a horn clause logic to answer queries such as whether or not the agent believes a given proposition. The interpretation of PGOAL and other modal terms makes use of the belief reasoner for implementing the

definition of these terms. For instance, the belief reasoner is used by the PGOAL interpreter to find (and set trigger on) its belief that the committed goal has been achieved or is impossible to achieve or is irrelevant.

Reasoning rules such as rules of rational action and rules for the domain of an application are not prescribed by the JI theory and so are left as declarative concepts. These rules are executed by a generic rule interpreter and it is the agent programmer's responsibility to ensure that the rule does not override the semantics of the terms from the JI theory. The fact that the semantics of the main modal constructs is built into the interpreter alleviates some of the potential problems. For instance, it is possible to come up with a rule that to tell a STAPLE agent to adopt a new PGOAL to achieve p whenever it believes that p is true. However, the agent will never end up adopting a new PGOAL in accordance with that rule (even if that rules fires) because it will discover the inconsistency before actually adopting the new commitment.

Communicative actions are declarative specified just like any other actions, and therefore, no special support is built into the interpreter for the communicative actions. The defeasible rules of communication are declaratively specified as reasoning rules. The effects, preconditions, etc. of the communicative actions specified as logical formulae that the STAPLE interpreter already knows how to handle.

Some aspects of the theory such as reasoning about intentions of other agents, maintenance goals, etc. have not yet been implemented and these are part of future research. Next, we briefly review the main contributions of this dissertation.

10.4 SUMMARY OF ACCOMPLISHMENTS

The present dissertation has made the following contributions to the field of multi-agent systems and dialogue.

1. Reasoning about teamwork & communication: It extended the prior work on agent communication languages based on the joint intention.

2. Interpreting logical constructs: It presented an agent programming language called STAPLE with built-in support for teamwork and multi-agent conversations and presented an implemented interpreter for STAPLE that directly executes agent specifications in a subset of modal logic, dynamic logic of actions, and temporal logic along with abstractions from the formal theory of teamwork and multi-agent conversations.

3. Group communication: It provided a formal semantics of group communication such that the individual communicative acts are a special case of the group communicative acts.

4. Persistent teams: It extended the joint intention theory to include teams that continue to exist even when the team membership changes.

5. Fault-tolerant Agent Architecture: It demonstrated the usefulness of persistent but dynamic teams by developing an agent architecture (called the Adaptive Agent Architecture or AAA) that implements a fault-tolerance specification based on this theory. AAA provided a test case for the STAPLE – brokers written in STAPLE must be able to recreate the same fault tolerant behavior as that of AAA brokers when the logical specification of that behavior is given to the STAPLE brokers.

6. Testing & Verification: It verified that STAPLE does in fact satisfy the above requirements. First, the specification of a fault-tolerant agent architecture (AAA) that is robust to sudden broker unavailability is provided to brokers written in STAPLE. The resulting STAPLE-based multi-agent system is shown to duplicate that fault-tolerant behavior. Second, STAPLE agents are shown to exhibit correct collaborative behavior in a simulated game that involves human-agent collaboration.

7. Multi-agent conversations: It introduces a technique to specify formal semantics of conversation protocols within the framework of the joint intention theory by introducing a notion of landmarks. This technique allows treatment of conversation protocols as joint actions similar to that in natural language dialogue, and enables formal proofs about correctness of a conversation protocol with respect to what its design goals.

8. Dialogue as joint action: The STAPLE agents presented in this dissertation demonstrated the following interesting dialogue behavior as a result of treating dialogue as joint actions:

    - If agents do not mutually believe that each of them can observe the actions and objects of interest in the world, then they communicate explicitly, otherwise explicit communication is absent.
    - If an agent has unbound variables, then it initiates the required dialogue to resolve its value.

- If an agent believes that another agent knows an answer, then it engages into a dialogue with that agent to find out the answer.
- If an agent believes that somebody else can do the action that it wanted done, then it establishes a team for that purpose by way of communication.
- Dialogue to establish mutual belief that a joint action is done, is impossible, or is irrelevant follows automatically.
- The above behavior can be chained to elicit even more interesting behavior.
- A sub-dialogue gets started in the middle of joint action execution to find the truth value of a proposition if it is the precondition for doing a part of the jointly committed action.

In short, dissertation demonstrates the feasibility of integrating the JI Theory, semantics of communicative acts and belief reasoning using a logic-based declarative language to obtain team and communicative behavior automatically, without having to program this behavior explicitly. The examples in this dissertation were created by encoding the initial conditions and stipulating the joint plan, from which team and communicative behavior followed automatically. There was no necessity to indicate when a question should be raised, when information should be shared etc. The plan and action library built into the STAPLE interpreter enables the agents to exhibit team-oriented dialogue by interpreting the constructs of joint intention theory along with first principles reasoning over a formal semantics of communicative acts based on that theory. This research shows that formal semantics of communicative acts can be fruitfully employed for inter-agent dialogue. Next, we discuss some of the important future research in the direction of this dissertation.

10.5 FUTURE WORK

The semantics of group communication in terms of "whoever" as presented in Chapter 5 is not supported in the present version of STAPLE interpreter and its implementation is an important future work in the direction of this dissertation. The research on semantics of group conversation protocols and its support in STAPLE is also left for future work. However, preliminary work on this topic can be found in [96].

This dissertation presented a formal semantics of maintenance goal and teams based on joint maintenance commitments. These concepts were used in AAA to specify a different kind of robust behavior wherein a broker team is able to maintain a specified number of

brokers for load balancing, fault-tolerance and other reasons. Implementation of constructs for maintenance goals and the corresponding joint maintenance commitments in STAPLE is another important future work.

As discussed in Chapter 8, the current version of STAPLE only supports fully specified conversation protocols. The support for full fledged conversation protocols in terms of joint action expressions as well as support for protocol families in terms of landmark expressions is another future enhancement to STAPLE.

The STAPLE interpreter needs to be enhanced and streamlined to support deadlines, and to optimize goals and tasks via scheduling. One of the important future works is to enhance STAPLE to enable its use as a generic dialogue engine. By virtue of implementing joint intention theory, we have seen that STAPLE agents automatically demonstrate confirmations, dialogue about referring, and other interesting aspects of dialogue. The implementation for PRS style planning techniques in STAPLE enables STAPLE agents to plan speech acts automatically to a certain extent. The main missing elements in STAPLE for supporting full-fledged dialogue are components for plan recognition, reasoning about intentions of other agents, and explicit default reasoning. Other enhancements for the dialogue engine include support for stochastic reasoning and multi-modal dialogue in the STAPLE interpreter. The programming language itself needs to be modified to add the supporting constructs for these enhancements.

We believe that treatment of roles and responsibilities in teams, organizations, and institutions is needed for a better understanding of what happens in group-communication in these complex groups and is part of future work. The concept of roles and responsibilities needs to be an integral part of the logic at the level of primitives. Ultimately, it is the agents that instantiate a role that make commitments and honor those commitments on behalf of the role. So there is a level of indirection – the commitments of a role have to be discharged by whoever is filling in that role. Moreover, instead of looking for agents with a particular capability, one may want to look for roles that have the capability, authority, and permission for doing a certain task. This would continue to work if the agent(s) instantiating a role change with time. Furthermore, roles are also important from a robustness perspective (along with groups that are required for persistent and dynamic teams) by allowing other capable agent(s) to fulfill the role(s) occupied by an agent that is no longer available. This dissertation does not explicitly address the notion of roles in the framework of teamwork and communication and it is left as an important direction for future work.

10.6 SUMMARY

The present dissertation demonstrated the feasibility of using a logic-based declarative interpreted language to program teams of autonomous agents. These agents exhibit correct team and communicative behavior without having to program explicitly. This dissertation extended an existing formal theory of teamwork (Joint Intention Theory) by specifying a formal semantics of multi-agent communication based on that theory along with other aspects of teamwork such as support for persistent and dynamic teams. Thereafter, it presented a domain independent agent programming language called STAPLE with built-in support for teamwork and multi-agent conversations based on these theoretical contributions. The usefulness of a declarative language such as STAPLE for programming teams of autonomous agents was demonstrated by showing that correct team and communicative behavior follow from agent specifications in two different domains without programming those behaviors in every possible situation. First, it was shown that brokers written in STAPLE demonstrate the same fault-tolerance behavior as that of brokers in a fault tolerant agent architecture (the AAA) by giving the logical specification of that behavior to STAPLE brokers. It was further demonstrated that STAPLE agents automatically engage in useful dialogue behavior required for the task at hand under different conditions of initial beliefs and agent capabilities in a simulated game that involved human agent collaboration. Future work in the direction of this dissertation includes full support for the theoretical concepts introduced in this dissertation into STAPLE, support for hybrid statistical-logical belief reasoning, explicit support for temporal concepts such as deadlines, and support for roles, responsibilities and organizations.

Bibliography

[1] Allen, J. F., and Perrault, C. R. Analyzing intention in dialogues. *Artificial Intelligence 15*, 3 (1980), 143–178.

[2] Appelt, D. E. Planning english referring expressions. In *Lecture Notes In Computer Science*, vol. 178. Elsevier Science, Essex, UK, 1985, pp. 1–33.

[3] Arisha, K., Eiter, T., Kraus, S., Ozcan, F., Ross, R., and Subrahmanian, V. Impact: Interactive maryland platform for agents collaborating together. *IEEE Intelligent Systems 14*, 2 (1998), 64–72.

[4] Bach, K., and Harnish, R. M. *Linguistic Communication and Speech Acts*. MIT Press, 1979.

[5] Bauer, B., Mller, J. P., and Odell, J. An extension of UML by protocols for multiagent interaction. In *Proceedings of the Fourth International Conference on MultiAgent Systems* (Boston, MA, USA, 2000), IEEE Press, pp. 207–214.

[6] Bernstein, P. A., and Newcomer, E. High availability, Chapter 7. In *Principles of Transaction Processing*. Morgan Kaufmann Publishers, Inc., San Francisco, CA, USA, 1997.

[7] Birman, K. P. Part III: Reliable distributed computing, Chapters 12-26.

[8] Bradshaw, J. M., Dutfield, S., Benoit, P., and Wooley, J. D. Kaos: Toward an industrial-strength open distributed agent architecture. In *Software Agents*, J. M. Bradshaw, Ed. AAAI/MIT Press, 1997.

[9] Brainov, S., and Sandholm, T. Reasoning about others: Representing and processing infinite belief hierarchies. In *Proceedings of the Fourth International Conference on Multi-Agent Systems* (Boston, MA, USA, 2000), IEEE Press, pp. 71–78.

[10] Bratman, M. *Intentions, Plans, and Practical Reason*. Harvard University Press, Cambridge, MA, 1987.

[11] Bratman, M. E. What is intention? In *Intentions in Communication*, P. R. Cohen, J. Morgan, and M. E. Pollack, Eds. MIT Press, 1990, pp. 15–33.

[12] Bratman, M. E., Israel, D. J., and Pollack, M. E. Plan and resource bounded practical reasoning. *Computational Intelligence 4*, 4 (1988), 349–355.

[13] Breazeal, C., Hoffman, G., and Lockerd, A. Teaching and working with robots as a collaboration. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multi-Agent Systems* (New York, 2004), ACM Press, pp. 1030–1037.

[14] Bretier, P., and Sadek, M. D. A rational agent as the kernel of a cooperative spoken dialogue system: Implementing a logical theory of interaction. In *Intelligent agents III - ECAI '96: Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages (ATAL)*, J. P. Muller, M. J. Wooldridge, and N. R. Jennings, Eds., Lecture Notes In Artificial Intelligence. Springer-Verlag, London, UK, 1996, pp. 189–203.

[15] Busetta, P., Dona, A., and Nori, M. Channeled multicast for group communications. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems* (Bologna, Italy, 2002), pp. 1280–1287.

[16] Busetta, P., Merzi, M., Ross, R., and Legras, F. Intra-role coordination using group communication: A preliminary report. In *Advances in Agent Communication*, F. Dignum, Ed., vol. 2922 of *Lecture Notes In Artificial Intelligence*. Springer Verlag, 2004, pp. 231–253.

[17] Castelfranchi, C. Commitments: from individual intentions to groups and organizations. In *Proceedings of the First International Conference on Multi-Agent Systems* (1995), pp. 41–48.

[18] Castelfranchi, C. Understanding the functions of norms in social groups through simulation. In *Artificial Societies*, N. Gilbert and R. Conte, Eds. UCL Press, London, 1995.

[19] Chaib-draa, B., and Vanderveken, D. Agent communication language: Towards a semantic based on success, satisfaction and recursion. In *Intelligent Agents V (ATAL-98)*, A. Rao, M. P. Singh, and J. P. Mueller, Eds., vol. 1555 of *Lecture Notes In Artificial Intelligence*. Springer-Verlag, Berlin, 1999, pp. 362–378.

[20] Chen, L., and Avizienis, A. N-version programming: A fault-tolerance approach to reliablity of software operation. In *Proceedings of the 8th Annual International Conference on Fault-Tolerant Computing* (Toulouse, France, 1978), pp. 3–9.

[21] Clark, H. H., and Wilkes-Gibbs, D. Referring as a collaborative process. *Cognition 22* (1986), 1–39.

[22] Cohen, P. R. *On Knowing What to Say: Planning Speech Acts.* PhD thesis, University of Toronto, 1978.

[23] Cohen, P. R., Cheyer, A., Wang, M., and Baeg, S. C. An open agent architecture. In *Proceedings of the AAAI Spring Symposium: Software Agents* (Menlo Park, CA, 1994), pp. 1–8.

[24] Cohen, P. R., Johnston, M., McGee, D. R., Oviatt, S. L., Pittman, J., Smith, I. A., Chen, L., and Clow, J. Quickset: Multimodal interaction for distributed applications. In *Proceedings of the Fifth International Multimedia Conference* (1997), ACM Press, pp. 31–40.

[25] Cohen, P. R., and Levesque, H. J. Intention is choice with commitment. *Artificial Intelligence 42*, 2-3 (1990), 213–261.

[26] Cohen, P. R., and Levesque, H. J. Performatives in a rationally based speech act theory. In *Proceedings of the 28th Annual Meeting of the Association for Computational Linguistics* (1990), pp. 79–88.

[27] Cohen, P. R., and Levesque, H. J. Rational interaction as the basis for communication. In *Intentions in Communication*, P. R. Cohen, J. Morgan, and M. E. Pollack, Eds., System Development Foundation Benchmark Series. MIT Press, Cambridge, MA, 1990, pp. 221–256.

[28] Cohen, P. R., and Levesque, H. J. Confirmations and joint action. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence* (Sydney, Australia, 1991), Morgan Kaufmann Publishers, San Francisco, USA, pp. 951–957.

[29] Cohen, P. R., and Levesque, H. J. Teamwork. *Nous 25*, 4 (1991), 487–512.

[30] Cohen, P. R., and Levesque, H. J. Communicative actions for artificial agents. In *Proceedings of the First International Conference on Multi-Agent Systems* (San Francisco, USA, 1995), V. Lesser, Ed., AAAI Press, pp. 65–72.

[31] Cohen, P. R., Levesque, H. J., Nunes, J. H. T., and Oviatt, S. L. Task-oriented dialogue as a consequence of joint activity. In *Proceedings of the Pacific Rim International Conference on Artificial Intelligence* (Nagoya, Japan, 1990), Morgan Kaufmann Publishers, Inc., pp. 203–208.

[32] Cohen, P. R., Morgan, J., and Pollack, M. E. *Intentions in Communication.* System Development Foundation Benchmark Series. MIT Press, 1990.

[33] Cohen, P. R., and Perrault, C. R. Elements of a plan-based theory of speech acts. *Cognitive Science 3*, 3 (1979), 177–212.

[34] CORBA. Common object request broker architecture. `http://www.omg.org`, 1997.

[35] Cost, R. S., Chen, Y., Finin, T., Labrou, Y., and Peng, Y. Modeling agent conversations with colored petri nets. In *Proceedings of the Workshop on Agent Conversation Policies at Third International Conference on Autonomous Agents (Agents-99)* (Seattle, WA, USA, 1999), pp. 59–66.

[36] Dastani, M., Riemsdijk, B. v., Dignum, F., and Meyer, J. J. A programming language for cognitive agents: Goal directed 3APL. In *Proceedings of the First Workshop on Programming Multiagent Systems: Languages, frameworks, techniques, and tools (ProMAS-03)* (Melbourne, Australia, 2003), pp. 111–130.

[37] Decker, K., Williamson, M., and Sycara, K. Matchmaking and brokering. In *Proceedings of the Second International Conference on Multi-agent Systems* (Kyoto, Japan, 1996), AAAI Press.

[38] Dennett, D. C. *The Intentional Stance.* Bradford Books / MIT Press, 1987.

[39] Dignum, F., Dunin-Keplicz, B., and Verbrugge, R. Dialogue in team formation. In *Issues in Agent Communication*, F. Dignum and M. Greaves, Eds., vol. 1916. Springer Verlag, 2000, pp. 264–280.

[40] Elio, R., and Haddadi, A. On abstract models and conversation policies. In *Issues in Agent Communication*, F. Dignum and M. Greaves, Eds., vol. 1916. Springer Verlag, 2000, pp. 301–313.

[41] FIPA. Agent communication language specifications. `http://www.fipa.org`, 2000.

[42] Fisher, M. Representing and executing agent-based systems. In *Proceedings of the First International Workshop on Agent Theories, Architectures, and Languages* (Amsterdam, The Netherlands, 1994), pp. 307–323.

[43] Georgeff, M. P., and Lansky, A. L. Procedural knowledge. *IEEE Special Issue on Knowledge Representation 74*, 10 (1986), 1383–1398.

[44] Giacomo, G. D., Lesprance, Y., and Levesque, H. J. Congolog, a concurrent programming language based on the situation calculus. *Artificial Intelligence 121*, 1-2 (2000), 109–169.

[45] Grosz, B., and Kraus, S. Collaborative plans for group activities. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence* (Chambery, France, 1993), vol. 1, pp. 367–373.

[46] Grosz, B. J., and Sidner, C. L. Plans for discourse. In *Intentions in Communication*, P. R. Cohen, J. Morgan, and M. E. Pollack, Eds. MIT Press, Cambridge, MA, 1990, pp. 417–444.

[47] Hägg, S. A sentinel approach to fault handling in multi-agent systems. In *Revised Papers from the Second Australian Workshop on Distributed Artificial Intelligence: Multi-Agent Systems: Methodologies and Applications*, vol. 1286 of *Lecture Notes In Computer Science*. Springer-Verlag, London, UK, 1997, pp. 181–195.

[48] Halpern, J. Y., and Moses, Y. Knowldege and common knowledge in a distributed environment. *Journal of the ACM 37*, 3 (1990), 549–587.

[49] Hindriks, K. V., Boer, F. S. d., Hoek, W. v. d., and Meyer, J.-J. C. Agent programming in 3APL. *Autonomous Agents and Multi-Agent Systems 2*, 4 (1999), 357–401.

[50] Hindriks, K. V., Boer, F. S. d., Hoek, W. v. d., and Meyer, J.-J. C. Agent programming with declarative goals. In *Proceedings of the 7th International Workshop on Intelligent Agents VII. Agent Theories Architectures and Languages*, vol. 1986 of *Lecture Notes In Computer Science*. Springer-Verlag, London, UK, 2000, pp. 228–243.

[51] Huber, M. J. JAM: A BDI-theoretic mobile agent architecture. In *Proceedings of the Third International Conference on Autonomous Agents (Agents '99)* (Seattle, WA, USA, 1999), pp. 236–243.

[52] Jennings, N. R. Controlling cooperative problem solving in industrial multi-agent systems using joint intentions. *Artificial Intelligence 75*, 2 (1995), 195–240.

[53] Jeon, H., Petrie, C., and Cutkosky, M. R. Jatlite: A java agent infrastructure with message routing. *IEEE Internet Computing 4*, 2 (2000), 87–96.

[54] Johnson-Laird, P. N. Mutual ignorance: Comments on clark and carlson's paper. In *Mutual Knowledge*, N. V. Smith, Ed. Academic Press, London, 1982, pp. 40–45.

[55] Kaminka, G. A., and Tambe, M. What is wrong with us? improving robustness through social diagnosis. In *Proceedings of the 15th National Conference on Artificial Intelligence* (Madison, Wisconsin, 1998), AAAI Press, pp. 97–104.

[56] Katagiri, Y. Belief coordination by default. In *Proceedings of the Second International Conference on Multiagent Systems* (Kyoto, Japan, 1996), MIT Press, pp. 142–149.

[57] Kinny, D., Ljungberg, M., Rao, A. S., Sonenberg, E. A., Tidhar, G., and Werner, E. Planned team activity. In *MAAMAW-92: Selected papers from the 4th European Workshop on on Modelling Autonomous Agents in a Multi-Agent World, Artificial Social Systems*, vol. 830 of *Lecture Notes In Computer Science*. Springer-Verlag, London, UK, 1994, pp. 227–256.

[58] Klein, M., and Dellarocas, C. Exception handling in agent systems. In *Proceedings of the Third International Conference on Autonomous Agents (Agents-99)* (Seattle, USA, 1999), pp. 62–68.

[59] Konolige, K. *A Deduction Model of Belief*. Research Notes in Artificial Intelligence. Morgan Kaufmann, San Mateo, California, 1986.

[60] Kumar, S. CHCC-Prolog. `http://www.cse.ogi.edu/CHCC/Agents/prolog.html`, 2002.

[61] Kumar, S., and Cohen, P. R. Towards a fault-tolerant multi-agent system architecture. In *Proceedings of the Fourth International Conference on Autonomous Agents (Agents 2000)* (Barcelona, Spain, 2000), ACM Press, pp. 459–466.

[62] Kumar, S., Cohen, P. R., and Huber, M. J. Direct execution of team specification in staple (short paper). In *Proceedings of the First International Joint Conference on Autonomous Agents and Multi-Agent Systems* (Bologna, Italy, 2002), pp. 567–568.

[63] Kumar, S., Cohen, P. R., and Levesque, H. J. The adaptive agent architecture: Achieving fault-tolerance using persistent broker teams. In *Proceedings of the Fourth International Conference on Multi-Agent Systems* (Boston, MA, USA, 2000), IEEE Press, pp. 159–166.

[64] Kumar, S., Huber, M. J., and Cohen, P. R. Representing and executing protocols as joint actions. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multi-Agent Systems* (Bologna, Italy, 2002), pp. 543–550.

[65] Kumar, S., Huber, M. J., Cohen, P. R., and McGee, D. R. Toward a formalism for conversation protocols using joint intention theory. *Computational Intelligence (Special Issue on Agent Communication Language) 18*, 2 (2002), 174–228.

[66] Kumar, S., Huber, M. J., McGee, D. R., Cohen, P. R., and Levesque, H. J. Semantics of agent communication languages for group interaction. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence* (Austin, Texas, USA, 2000), AAAI Press, pp. 42–47.

[67] Labrou, Y., and Finin, T. A proposal for a new kqml specification. Technical Report TR CS-97-03, Computer Science and Electrical Engineering Department, UMBC, 1997.

[68] Labrou, Y., and Finin, T. Semantics and conversations for an agent communication language. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence* (Nagoya, Japan, 1997), pp. 584–591.

[69] Laird, J. E., Newell, A., and Rosenbloom, P. S. Soar: an architecture for general intelligence. *Artificial Intelligence 33*, 1 (1987), 1–64.

[70] Levesque, H. J., Cohen, P. R., and Nunes, J. H. T. On acting together. In *Proceedings of the Eighth National Conference on Artificial Intelligence* (Boston, MA, USA, 1990), pp. 94–99.

[71] Lomet, D., and Weikum, G. Efficient transparent application recovery in client-server information systems. In *Proceedings of the SIGMOD '98: Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data* (Seattle, USA, 1998), ACM Press, pp. 460–471.

[72] Martin, D. L., Cheyer, A. J., and Moran, D. B. The open agent architecture: A framework for building distributed software systems. *Applied Artificial Intelligence 13*, 1-2 (1999), 91–128.

[73] Minton, S., Bresina, J., and Drummond, M. Total order and partial order planning: a comparative analysis. *Journal of Artificial Intelligence Research 2* (1994), 227–262.

[74] Morley, D., and Myers, K. L. The SPARK agent framework. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multi-Agent Systems* (New York, USA, 2004), ACM Press, pp. 714–721.

[75] Nair, R., and Tambe, M. Hybrid BDI-POMDP framework for multiagent teaming. *Journal of Artificial Intelligence Research (JAIR) 23* (2005), 367–420.

[76] Neiger, G. Knowledge consistency: A useful suspension of disbelief. In *Proceedings of the Second Conference on Theoretical Aspects of Reasoning about Knowledge* (1988), M. Y. Vardi, Ed., Morgan Kaufmann, San Francisco, USA, pp. 295–308.

[77] Neiger, G. Simplifying the design of knowledge-based algorithms using knowledge consistency. *Information and Computation 119*, 2 (1995), 283–293.

[78] Nodine, M., Perry, B., and Unruh, A. Experience with the infosleuth agent architecture. In *Proceedings of the AAAI-98 Workshop on Software Tools for Developing Agents* (1998).

[79] Norman, T. J., and Reed, C. A. Group delegation and responsibility. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'02)* (Bologna, Italy, 2002), pp. 491–498.

[80] O'Hare, G. M. P., and Jennings, N. R. *Foundations of Distributed Artificial Intelligence*. John Wiley and Sons, New York, 1996.

[81] Ortiz, C., and Vincent, R. Realtime dynamic coalition formation. In *Proceedings of the AAAI Workshop on Coalition Formation in Dynamic Multiagent Environments* (2002).

[82] Parunak, H. V. D. Visualizing agent conversations: Using enhanced dooley graphs for agent design and analysis. In *Proceedings of the Second International Conference on Multi-agent Systems* (1996), pp. 275–282.

[83] Perrault, C. R. An application of default logic to speech act theory. In *Intentions in Communication*, P. R. Cohen, J. Morgan, and M. E. Pollack, Eds. MIT Press, Cambridge, MA, 1990, pp. 161–185.

[84] Pitt, J., Kamara, L., and Artikis, A. Interaction patterns and observable commitments in a mutli-agent trading scenario. In *Proceedings of the Fifth International Conference on Autonomous Agents (Agents 2001)* (Montreal, Quebec, Canada, 2001), ACM Press, pp. 481–488.

[85] Pitt, J., and Mamdani, A. Designing agent communication languages for multi-agent systems. In *Proceedings of the Modeling Autonomous Agents in a Multi-Agent World (MAAMAW)* (Valencia, Spain, 1999), pp. 102–114.

[86] Pitt, J., and Mamdani, A. A protocol-based semantics for an agent communication language. In *Proceedings of the Sixteenth International Joint Conference on Articial Intelligence* (Stockholm, Sweden, 1999), pp. 486–491.

[87] Pynadath, D. V., and Tambe, M. The communicative multiagent team decision problem: Analyzing teamwork theories and models. *Journal of Artificial Intelligence Research (JAIR) 16* (2002), 389–423.

[88] Pynadath, D. V., Tambe, M., Chauvat, N., and Cavedon, L. Toward team-oriented programming. In *Proceedings of the 6th International Workshop on Intelligent Agents VI, Agent Theories, Architectures, and Languages (ATAL'99)* (1999), pp. 233–247.

[89] Rao, A. S. Agentspeak(L): BDI agents speak out in a logical computable language. In *Lecture Notes In Artificial Intelligence*, vol. 1038. Springer-Verlag, 1996, pp. 42–55.

[90] Rao, A. S., and Georgeff, M. P. Modeling rational agents within a BDI architecture. In *Proceedings of the Second International Conference on Knowledge Representation and Reasoning* (1991), pp. 473–484.

[91] Rao, A. S., Georgeff, M. P., and Sonenberg, E. A. Social plans: A preliminary report. In *Decentralized AI 3*, E. Werner and Y. Demazeau, Eds. Elsevier Science Publishers, Amsterdam, 1992, pp. 57–76.

[92] Reiter, R. The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result for goal regression. In *Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy*, V. Lifschitz, Ed. Academic Press, San Diego, CA, 1991, pp. 359–380.

[93] Rich, C., Sidner, C. L., and Lesh, N. B. Collagen: Applying collaborative discourse theory to human-computer interaction. *Artificial Intelligence Magazine 22*, 4 (2001), 15–25.

[94] Rosenschein, S. Plan synthesis: a logical approach. In *Proceedings of the 8th International Joint Conference on Artificial Intelligence* (1981), Academic Press, pp. 359–380.

[95] Rossi, S., and Busetta, P. Towards monitoring of group interactions and social roles via overhearing. In *Proceedings of the Eight International Workshop on "Cooperative Information Agents (CIA-04)*, vol. 3191 of *Lecture Notes in Computer Science*. Erfurt, Germany, 2004, pp. 47–61.

[96] Rossi, S., Kumar, S., and Cohen, P. R. Distributive and collective readings in group protocols. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence* (Edinburgh, Scotland, 2005), pp. 971–976.

[97] Sadek, M. D. Dialogue acts are rational plans. In *Proceedings of the ESCA/ETRW Workshop on the structure of multimodal dialogue* (Maratea, Italy, 1991), pp. 1–29.

[98] Sadek, M. D., Bretier, P., and Panaget, F. ARTIMIS: Natural language meets rational agency. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence* (Nagoya, Japan, 1997), pp. 1030–1035.

[99] Sandholm, T. W., and Lesser, V. R. Advantages of a leveled commitment contracting protocol. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence* (Portland, OR, 1996), pp. 126–133.

[100] Searle, J. *Speech Acts*. Cambridge University Press, 1969.

[101] Searle, J. R. *Intentionality: An Essay in the Philosophy of Mind*. Cambridge University Press, 1983.

[102] Searle, J. R. Collective intentions and actions. In *Intentions in Communication.*, P. R. Cohen, J. Morgan, and M. E. Pollack, Eds. MIT Press, Cambridge, Ma., 1990, pp. 401–416.

[103] Shoham, Y. Agent-oriented programming. *Artificial Intelligence 60*, 1 (1993), 51–92.

[104] Singh, M. P. The intentions of teams: Team structure, endodeixis, and exodeixis. In *Proceedings of the 13th European Conference on Artificial Intelligence* (Brighton, UK, 1998), Wiley, pp. 303–307.

[105] Singh, M. P. A social semantics for agent communication languages. In *ssues in Agent Communication*, F. Dignum and M. Greaves, Eds., vol. 1916. Springer Verlag, 2000, pp. 31–45.

[106] Smith, I. A., and Cohen, P. R. Toward a semantics for an agent communications language based on speech-acts. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence* (Portland, Oregon, 1996), AAAI Press, pp. 24–31.

[107] Smith, I. A., Cohen, P. R., Bradshaw, J. M., Greaves, M., and Holmback, H. Designing conversation policies using joint intention theory. In *Proceedings of the Third International Conference on Multi Agent Systems* (Paris, France, 1998), IEEE Press, pp. 269–276.

[108] Subrahmanian, V., Dix, J., Eiter, T., Bonatti, P., Kraus, S., Ozcan, F., and Ross, R. *Heterogeneous Agent Systems.* MIT Press, 2000.

[109] Tambe, M. Agent architectures for flexible, practical teamwork. In *Proceedings of the 14th National Conference on Aritificial Intelligence* (Providence, Rhode Island, 1997), AAAI Press, pp. 22–28.

[110] Tambe, M. Towards flexible teamwork. *Journal of Artificial Intelligence Research 7* (1997), 83–124.

[111] Tambe, M., and Zhang, W. Towards flexible teamwork in persistent teams: Extended report. *Journal of Autonomous Agents and Multi-agent Systems 3* (2000), 159–183.

[112] Tuomela, R., and Miller, K. We-intentions. *Philosophical Studies 53* (1988), 367–389.

[113] Vongkasem, L., and Chaib-draa, B. ACL as a joint project between participants: A preliminary report. In *Issues in Agent Communication*, F. Dignum and M. Greaves, Eds., vol. 1916 of *Lecture Notes In Artificial Intelligence*. Springer Verlag, 2000, pp. 235–248.

[114] Werner, E. Cooperating agents: A unified theory of communication and social structure. In *Distributed Artificial Intelligence*, L. Gasser and M. Huhns, Eds., vol. 2. Morgan Kaufmann, Publishers, Inc., 1990, pp. 3–36.

[115] Winograd, T., and Flores, F. *Understanding Computers and Cognition: A New Foundation for Design.* Addison-Wesley, Reading, MA, 1987.

[116] Wong, H. C., and Sycara, K. A taxonomy of middle-agents for the internet. In *Proceedings of the Fourth International Conference on MultiAgent Systems* (Boston, MA, USA, 2000), pp. 465–466.

[117] Wooldridge, M. J. *The Logical Modelling of Computational Multi-Agent Systems.* PhD thesis, UMIST, 1992.

[118] Wooldridge, M. J. *Reasoning About Rational Agents.* MIT Press, 2000.

[119] Wooldridge, M. J., and Jennings, N. R. Formalizing the cooperative problem solving process. In *Proceedings of the Thirteenth International Workshop on Distributed Artificial Intelligence* (Lake Quinalt, WA, 1994), pp. 403–417.

[120] Wooldridge, M. J., and Jennings, N. R. Ecai-94: Agent theories, architectures, and languages: A survey. In *Proceedings of the ECAI Workshop on Agent, Theories, Architectures, and Languages* (Amsterdam, The Netherlands, 1995), pp. 1–39.

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

[122] Yadgar, O., Kraus, S., and Ortiz, C. Scaling up distributed sensor networks: cooperative large-scale mobile-agent organizations. In *Distributed Sensor Networks: a multiagent perspective*. Kluwer publishing, 2003, pp. 185–218.

[123] Yolum, P., and Singh, M. P. Commitment-based enhancement of e-commerce protocols. In *Proceedings of the IEEE 9th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises* (National Institute of Standards and Technology (NIST), Gaithersburg, Maryland, USA, 2000), IEEE Press, pp. 278–283.

Appendix A

STAPLE: The Language and its Operational Semantics

A.1 OVERVIEW

This dissertation introduced an agent programming language called STAPLE using concrete examples of agent programs written in STAPLE (Chapter 3) and it presented an implemented STAPLE interpreter (Chapter 4). The standard practice in literature [49, 89] is to present an agent programming language at a sufficiently high level along with its formal operational semantics by abstracting away from the myriad implementation choices and practical compromises inevitable in such endeavors. Joint intention theory provides the axiomatic/logical semantics of STAPLE, and an agent written in STAPLE achieves team and communicative behavior according to JI theory. An operational semantics of STAPLE would enable showing its formal connection to the joint intention theory, and at the same time would allow one to see how an implementation of the language deviates from its formal specification.

We present a formal account of a subset of STAPLE and its operational semantics in this appendix in a manner that will be useful for specifying its operational semantics. As such, the language definition may appear to be a repetition of the constructs from joint intention theory using a different formalism. For most part, the operational semantics itself formalizes the results that follow from the definitions and theorems in this dissertation in a way that can be implemented in a STAPLE interpreter. An interpreter for STAPLE has been implemented based on the language definition and its operational semantics presented in this appendix. It is important to note that the formal properties and connection to JI theory discussed in this appendix refer to the language itself, and not to any particular implementation.

We first present a simplified version of STAPLE that supports programming single agents and then progressively modify it to get the final version of STAPLE that supports joint activity as per the JI theory. The simplified language and its (single agent) constructs are presented in Section A.2, relevant parts of its operational semantics are discussed in Section A.3, and the theory of basic actions in STAPLE is presented in Section A.4. The language presented thus far is modified in Section A.5 to support joint actions, groups, and communicative acts. Section A.6 presents practical optimizations in STAPLE, and we conclude in Section A.7 with a discussion on relationship between operational semantics presented in this appendix and the joint intention theory.

A.2 LANGUAGE DEFINITION

The mental state of STAPLE agents consists of formulae in a first order modal language $\mathcal{L}$ with equality as well as operators for propositional attitudes, temporal properties, and event sequences. We first introduce the first order version of this language, define actions and action expressions, and then add the modal operators before formally defining the mental state of a STAPLE agent.

Definition A.1. *The first order language* $\mathcal{L}$

Let Var be a set of countably infinite variables consisting of action variables ActionVar (elements a,b,$\cdots$,a1,a2,$\cdots$,b1,b2 $\cdots$), agent variables AgtVar (elements x,y,$\cdots$, x1,x2, $\cdots$,y1,y2,$\cdots$), and regular variables (elements i,j,$\cdots$,i1,i2,$\cdots$,j1,j2,$\cdots$). Let Const be the set of constants (names) in the domain including the symbols true, false, and self. Let Agt $\subseteq$ Const be the set of agents in the domain.

Let Pred be the set of n-ary predicate symbols, Func be the set of function symbols, and let Term be the set of terms built from Const, Func, and Var. The language $\mathcal{L}$ is then defined as follows.

1. if p $\in$ Pred and t1,$\cdots$,tn $\in$ Term, then p(t1,$\cdots$,tn) $\in \mathcal{L}$
2. if t1, t2 $\in$ Term, then t1=t2 $\in \mathcal{L}$
3. if $\varphi \in \mathcal{L}$, then $\neg\varphi \in \mathcal{L}$
4. if $\varphi \in \mathcal{L}$ and $\psi \in \mathcal{L}$, then $\varphi \wedge \psi \in \mathcal{L}$, and $\varphi \vee \psi \in \mathcal{L}$
5. if $\varphi \in \mathcal{L}$ and x $\in$ Var, then $\forall$ x φ(x) $\in \mathcal{L}$, and $\exists$x φ(x) $\in \mathcal{L}$

Definition A.2. *Singleton Action*

Let ActSym $\subseteq$ Func be the set of action symbols.
A singleton action act(t1,$\cdots$,tn) is defined by the tuple $\langle$x,φ,ψ,σ,$\chi\rangle$

where act $\in$ ActSym is the name of the action,

t1,$\cdots$,tn $\in$ Term are the parameters to the action,

x $\in$ AgtVar is the actor of the action,

φ, ψ, $\sigma \in \mathcal{L}$, and specify the precondition, the post-condition, and the context of the action.

$\chi \in \mathcal{L}$ is the "executable code" for the action

The context is a frame condition in that it is not changed by the action but it is also an enabling condition for the action - it must be true right before action execution and remain true throughout the execution of the action. We define three functions $pre(a)$, $post(a)$, and $context(a)$ that return φ, ψ, and σ respectively for the singleton action a.

Definition A.3. *Action Expression*

Let SAct be the set of singleton actions. An action expression AExp is then defined as follows.

1. if a $\in$ ActionVar, then a $\in$ AExp
2. if a $\in$ SAct, then a $\in$ AExp
3. if a1 $\in$ AExp and a2 $\in$ AExp, then
   (a) a1;a2 $\in$ AExp (action sequence)
   (b) a1|a2 $\in$ AExp (non-deterministic OR)
   (c) a1||a2 $\in$ AExp (concurrent actions)
   (d) a* $\in$ AExp (Infinite repetition)
4. if $\varphi \in \mathcal{L}$ then $\varphi?$ $\in$ AExp (Test action)

We augment the language $\mathcal{L}$ by adding modal operators pertaining to action expressions, commitments, and intentions.

Definition A.4. *Modal Language $\mathcal{L}$*

Let Agt be the set of agents, and let Actor = AgtVar ∪ Agt. The modal language $\mathcal{L}$ is then defined as follows.

1. if $\varphi \in \mathcal{L}$, then $\Diamond\varphi \in \mathcal{L}$, and $\Box\varphi \in \mathcal{L}$
2. if a ∈ AExp, then done(a) $\in \mathcal{L}$
3. if a ∈ AExp and x ∈ Actor, then done(x,a) $\in \mathcal{L}$
4. if $\varphi \in \mathcal{L}$ and x ∈ Actor, then bel(x, φ) $\in \mathcal{L}$
5. if $\varphi \in \mathcal{L}$ and x ∈ Actor, then goal(x, φ) $\in \mathcal{L}$
6. if φ,q $\in \mathcal{L}$ and x ∈ Actor, then pgoal(x,φ,q) $\in \mathcal{L}$
7. if q $\in \mathcal{L}$, a ∈ AExp, and x ∈ Actor, then intend(x,a,q) $\in \mathcal{L}$

The ground language $\mathcal{L}_G$ is a subset of the language that is used in the definition of an agents mental state.

Definition A.5. The Ground Language $\mathcal{L}_G$

Let Free(φ) denote the set of free variables in the formula $\varphi \in \mathcal{L}$. We first define a language $\mathcal{L}_G$ as consisting of ground formulae (i.e., sentences) from the language $\mathcal{L}$. $\mathcal{L}_G \subseteq \mathcal{L}$ s.t. if $\varphi \in \mathcal{L}$, and Free(φ) = $\emptyset$, then $\varphi \in \mathcal{L}_G$

Now, we are in a position to define the various components of an agent's mental state.

Definition A.6. Semantic Entailment ($\models$)

The definition of mental state that follows uses semantic entailment over sentences from $\mathcal{L}_G$. This entailment relationship is based on the logical semantics of the various mental state constructs (or modalities). This logical semantics is the same as the logic of joint intentions (Chapter 2). Any implementation of an interpreter for the language, however, is likely to use provability ($\vdash$) instead of semantic entailment. For example, a belief reasoner of such an interpreter would use the usual inference rules of first order logic along with axioms describing the properties of each modality and their relationship with each other. However, like other researchers [49, 50], we choose to use semantic entailment for the purpose of formally describing the abstract language. The implementation of STAPLE presented in the Chapter 4 uses provability instead of semantic entailment.

Definition A.7. *Mental State*
The mental state $\mathbb{M}$ of a STAPLE agent is defined by the tuple $\langle \mathcal{B}, \mathcal{G}, \mathcal{C}, \mathcal{I} \rangle$ where, $\mathcal{B}$ is the agent's belief base, $\mathcal{G}$ is goal base, $\mathcal{C}$ is the set of its internal commitments, and $\mathcal{I}$ contains the intentions of the agent. The details of each component of the mental state follow.

Beliefs **($\mathcal{B}$):** An agent's belief base consists of sentences from $\mathcal{L}_G$, that is, $\mathcal{B} \in \wp(\mathcal{L}_G)$ where $\wp(\mathcal{L}_G)$ denotes the power set of $\mathcal{L}_G$.
Goals **($\mathcal{G}$):** An agent's goal base consists of sentences from $\mathcal{L}_G$, that is, $\mathcal{G} \in \wp(\mathcal{L}_G)$.
Commitments **($\mathcal{C}$):** An agent's commitment base consists of tuples $\langle \varphi$,q$\rangle$ denoting that φ is the proposition that the agent is committed to achieving and this commitment is relative to some higher-level goal or context q. In other words,

$\mathcal{C} \subseteq \{\langle \varphi$,q$\rangle | \varphi \in \mathcal{L}$, q $\in \mathcal{L}\}$ with the following constraints:

1. $\mathcal{B} \cup \{\neg\varphi\} \not\models$ false, that is, φ must be consistent with the belief base.
2. $\mathcal{B} \not\models \varphi$, that is, φ must not be entailed by the belief base.
3. $\mathcal{B} \not\models \Box\neg\varphi$, that is, the impossibility of φ must not be entailed by the agent's belief base.
4. $\mathcal{B} \not\models \neg$q, that is, the belief base must not entail that the commitment is irrelevant[1].
5. $\mathcal{C} \not\models$ false, that is, the commitments must be consistent.
6. $\mathcal{I} \cup \{$pgoal(self,φ,q)$\} \not\models$ false, that is, this commitment must be consistent with the agent's intentions.

Intentions **($\mathcal{I}$):** An agent's intention base $\mathcal{I}$ consists of tuples $\langle$a,q$\rangle$ denoting that the agent intends to do action a relative to some higher-level goal or context q. The action a is either an action expression or a plan. Let φ,ψ,σ be the precondition, post-condition, and the context of the action.
$\mathcal{I} \subseteq \{\langle$a,q$\rangle |$ a $\in$ AExp or a $\in$ Plan , and q $\in \mathcal{L}\}$ with the following constraints:

1. $\mathcal{B} \not\models \psi$, that is, the effects of the action must not be entailed by the belief base.
2. $\mathcal{B} \not\models \sigma$, that is, the belief base must not entail that the context of the action is false.

[1]Strictly speaking, from the definition of PGOAL, an agent need not drop an irrelevant commitment. However, we make this simplifying assumption in STAPLE

3. $\mathcal{B} \not\models \Box\neg\text{done}(a)$ and $\mathcal{B} \not\models \Box\neg\varphi$ and $\mathcal{B} \not\models \Box\neg\sigma$, that is, the impossibility of doing a must not be entailed by the agent's belief base.

4. $\mathcal{B} \not\models \neg$q, that is, the belief base must not entail that the intention is irrelevant.

5. $\mathcal{I} \not\models$ false, that is, the intentions must be consistent.

6. $\mathcal{C} \cup \{\text{intend(self,a,q)}\} \not\models$ false, that is, this intention must be consistent with the agent's internal commitments.

From the above definition of the components of an agent's mental state, we see that the type of mental state $\mathcal{T}_M$ is given by $\wp(\mathcal{L}_G) \times \wp(\mathcal{L}_G) \times \wp(\mathcal{L}_G \times \mathcal{L}_G) \times \wp(\mathsf{AExp} \times \mathcal{L}_G)$, that is, it consists of a set of beliefs, a set of goals, a set of commitments, and a set of intentions.

Definition A.8. *Introspection / Effective Belief Base*

STAPLE agents are introspective about their mental attitudes. A consequence of this property is that the effective belief base used to test beliefs is larger than the actual belief base. The effective belief base $\mathcal{B}_E$ of STAPLE agents is defined as follows:

- if $\varphi \in \mathcal{B}$, then $\varphi \in \mathcal{B}_E$
- if $\langle\varphi$,q$\rangle \in \mathcal{C}$, then pgoal(self, φ,q) $\in \mathcal{B}_E$
- if $\langle$a,q$\rangle \in \mathcal{I}$, then intend(self, a,q) $\in \mathcal{B}_E$

Definition A.9. *Basic Actions*

These are primitive actions that are defined within STAPLE, form a part of every agent's capability, and they cannot be redefined or overridden by an agent programmer. Example of basic actions include *assert, retract, adopt, subgoal, drop,* and *execute_rule.* The semantics of some of these actions is discussed in the next section.

Definition A.10. *Rules for Rational Action*

Rules in a STAPLE program have two components: the premise and the action to execute if the premise is true. As such rules can be thought of as a special kind of conditional action and are represented as

$\psi \rightarrow a$

where, $\psi \in \mathcal{L}$ is the premise of the rule, and

$a \in \mathsf{AExp}$ is an expression such that every component of a is a *basic action* according to (A.9)

STAPLE defines a primitive action to execute rules. The semantics of rule execution is specified in A.13.

Next, we present the operational semantics of the language presented in this section.

A.3 OPERATIONAL SEMANTICS

As is customary in this field, we specify the operational semantics of STAPLE by means of a transition system. A transition system consists of a set of derivation rules that specify how to derive new possible computation steps from the computation steps and the conditions listed in their premise. We will use derivation rules of the form $\frac{C1,C2,\cdots,Cn}{S}$ where $C1$, $C2$, .., Cn are the conditions in the premise and S is the resulting computation step. Each computation step S changes the mental state of an agent and is specified as a transition from the current mental state to the next. The substitution that results from a computation step is explicitly specified as part of that step.

Definition A.11. *Substitution*

A substitution θ is a finite set of pairs (x_i,t_i) where $x_i \in \mathsf{Var}$ and $t_i \in \mathsf{Term}$, and $x_i \neq x_j$ for i $\neq$ j, and $x_i \notin$ Free(t_j) for all i and j. The domain of the substitution dom(θ) is the set $\{x_i |\ (x_i,t_i) \in \theta\}$. The substitution θ is said to be ground if Free(t)=$\emptyset$ for every (x,t) $\in \theta$.

Let η be an expression (a formula or an action exp.) and θ be a substitution. Then $\eta\theta$ denotes the expression obtained by simultaneously replacing all free variables x in η for which (x,t) $\in \theta$ by the corresponding term t.

Definition A.12. *Semantics of Action Execution*

Actions transform the mental state of agents besides possibly changing the state of the external world. Successful performance of an action not only satisfies the intention for that action but may also either satisfy or make impossible or irrelevant the other commitments and intentions of the agent. As such, the transformation of mental state by a singleton action a that is represented by act(x1,$\cdots$,xn) is expressed by a function $\mathcal{M}$ defined as follows.

$\mathcal{M}$(a, $\langle\mathcal{B},\mathcal{G},\mathcal{C},\mathcal{I}\rangle$) = $\langle\mathcal{B}',\mathcal{G},\mathcal{C}',\mathcal{I}'\rangle$ where,

$\mathcal{B}' = \mathcal{B} \cup \{post(a), \text{done}(a)\}$

$\mathcal{C}' = \mathcal{C}$ - $\{\langle\phi,\text{q}\rangle \in \mathcal{C} |\ \mathcal{B}' \models \phi$ or $\mathcal{B}' \models \Box\neg\phi$ or $\mathcal{B}' \models \neg$q$\}$

$\mathcal{I}' = \mathcal{I}$ - $\{\langle$b,p$\rangle \in \mathcal{I} |\ \mathcal{B}' \models post$(b) or $\mathcal{B}' \models \Box\neg post$(b) or $\mathcal{B}' \models \Box\neg$done(b) or $\mathcal{B}' \models \neg$p$\}$

Note that, $\langle a,\mathrm{q}\rangle \notin \mathcal{I}'$ because the post condition of the action that was just performed is entailed by the new belief base.

Definition A.13. *Semantics of Rule execution*

From Definition A.10, a rule ρ is a structure of the form $\psi \rightarrow a$ where a is a basic action. For the purpose of describing the operational semantics, it is frequently useful to talk about executing rules applicable to a given condition. This applicability condition $\phi \in \mathcal{L}$ is one of the conjuncts (usually the first conjunct) of the premise ψ and can be thought of as the "main premise" of the rule. The predefined basic action *execute_rule*(ϕ) combines both rule selection and rule execution. It first finds a rule ρ that has a conjunct φ in its premise such that φ unifies with ϕ, thereafter it unifies φ and ϕ and tests if the premise ψ is entailed by the effective belief base $\mathcal{B}_E$ and finally executes the rule action a if the test succeeds. Therefore, the exact transformation of mental state due to rule execution depends on the specifics of the rule that was executed. The semantics of rule execution is described by a function $\mathcal{R}$ of type: $\mathcal{L}_G \times \mathcal{L}_G \times \mathrm{AExp} \times \mathcal{T}_M \rightarrow \mathcal{T}_M$ where, $\mathcal{T}_M$ is the type of $\langle \mathcal{B},\mathcal{G},\mathcal{C},\mathcal{I}\rangle$. The result of rule execution is specified by the following derivation step where φ is the applicability criterion for the rule $\varphi \wedge \psi' \rightarrow a$.

$$\frac{\begin{array}{l}(\psi \rightarrow a) \in \mathrm{Rules}, \psi = \varphi \wedge \psi', \\ \phi \vDash \varphi\theta', \mathrm{dom}(\theta') = \mathrm{Free}(\varphi), \mathcal{B}_E \vDash \psi\theta'\theta, \mathrm{dom}(\theta) = \mathrm{Free}(\psi\theta'), \\ \mathcal{M}(a\theta'\theta, \langle \mathcal{B}, \mathcal{G}, \mathcal{C}, \mathcal{I}\rangle) = \langle \mathcal{B}', \mathcal{G}, \mathcal{C}', \mathcal{I}'\rangle\end{array}}{\mathcal{R}(\phi,\psi, a, \langle \mathcal{B}, \mathcal{G}, \mathcal{C}, \mathcal{I}\rangle) = \langle \mathcal{B}', \mathcal{G}', \mathcal{C}', \mathcal{I}'\rangle} \tag{A.1}$$

The above rule says that *execute_rule*(ϕ) works as follows: (i) if there is a rule $\psi \rightarrow a$ where $\psi = \varphi \wedge \psi'$ and (ii) if there exists a substitution θ' such that $\varphi\theta'$ (obtained by applying θ' to the first conjunct φ of the rule) is entailed by ϕ and the domain of the substitution θ' consists of the free variables in φ and (iii) if there exists a substitution θ that when applied to $\psi\theta'$ (obtained by applying the earlier substitution θ' to the premise ψ of the rule) such that $\psi\theta'\theta$ is entailed by the effective belief base and the domain of θ consists of free variables in $\psi\theta'$, and (iv) if $\mathcal{M}(a\theta'\theta, \langle \mathcal{B},\mathcal{G},\mathcal{C},\mathcal{I}\rangle) = \langle \mathcal{B}',\mathcal{G},\mathcal{C}',\mathcal{I}'\rangle$ that is, executing the action $a\theta'\theta$ (obtained by successively applying substitutions θ' and θ' to the action a of the rule $\psi \rightarrow a$) in mental state $\langle \mathcal{B},\mathcal{G},\mathcal{C},\mathcal{I}\rangle$ results in the mental state $\langle \mathcal{B}',\mathcal{G},\mathcal{C}',\mathcal{I}'\rangle$ then executing the rule $\psi \rightarrow a$ in the mental state $\langle \mathcal{B},\mathcal{G},\mathcal{C},\mathcal{I}\rangle$ results in the mental state $\langle \mathcal{B}',\mathcal{G},\mathcal{C}',\mathcal{I}'\rangle$ where $\langle \mathcal{B}',\mathcal{G},\mathcal{C}',\mathcal{I}'\rangle$ is specified in A.12. In the remainder of this appendix, we will use $\mathcal{R}(\phi,\psi,a,\langle \mathcal{B},\mathcal{G},\mathcal{C},\mathcal{I}\rangle)$ to denote the effect of executing a rule with applicability condition ϕ, premise ψ, and rule action a. Most of the time, we may not care what the premise and action of a rule are and we will denote its execution by $\mathcal{R}(\phi,\_,\_,\langle \mathcal{B},\mathcal{G},\mathcal{C},\mathcal{I}\rangle)$.

Definition A.14. *Interpreting Intention*

Intentions result in execution of two types of actions in STAPLE – test actions and singleton actions. A test action $\varphi?$ is always executed when an agent has the intention to do $\varphi?$. A test action checks whether an instance of φ is entailed by the agent's effective belief base $\mathcal{B}_E$, and if so, it returns a set of variable bindings θ such that $\text{dom}(\theta) = \text{Free}(\varphi)$. The substitution θ is empty if the test fails and non-empty otherwise. The execution of test actions is expressed by the following two computation steps – one for success and one for failure of the test action.

$$\text{(A.2)} \qquad \frac{\langle \varphi?, q\rangle \in \mathcal{I}, \mathcal{B}_E \vDash q\theta', \mathcal{B}_E \vDash \varphi\theta'\theta, \text{dom}(\theta) = \text{Free}(\varphi\theta')}{\langle \varphi?, \langle \mathcal{B}, \mathcal{G}, \mathcal{C}, \mathcal{I}\rangle\rangle \rightarrow_\theta \langle nil, \langle \mathcal{B}, \mathcal{G}, \mathcal{C}, \mathcal{I} - \{\langle \varphi?, \text{q}\rangle\}\rangle\rangle}$$

$$\text{(A.3)} \qquad \frac{\langle \varphi?, q\rangle \in \mathcal{I}, \mathcal{B}_E \vDash \text{q}\theta', \mathcal{B}_E \nvDash \varphi\theta'\theta; \text{for any } \theta}{\langle \varphi?, \langle \mathcal{B}, \mathcal{G}, \mathcal{C}, \mathcal{I}\rangle\rangle \rightarrow_\emptyset \langle nil, \langle \mathcal{B}, \mathcal{G}, \mathcal{C}, \mathcal{I}\rangle\rangle}$$

The derivation rule A.2 states that (i) if the agent has an intention to do the test action $\varphi?$ relative to q and (ii) if there is a substitution θ' such that $\text{q}\theta'$ is entailed by the agent's effective belief base $\mathcal{B}_E$ and (iii) if there is a substitution θ such that $\varphi\theta'\theta$ (the proposition obtained by successively applying substitutions θ' and θ to propostion φ to be tested) is entailed by the agent's effective belief base and domain of θ consists of free variables in $\varphi\theta'$ then executing the test action $\varphi?$ in the mental state $\langle\mathcal{B},\mathcal{G},\mathcal{C},\mathcal{I}\rangle$ returns the substitution θ and results in a new mental state where it does not have the intention to do the test action $\varphi?$. Furthermore, the agent does not have any more actions to execute in this execution step (i.e., nil action remains to be executed after executing the action $\varphi?$). The substitution θ returned by this execution step is indicated in the derivation rule as a subscript on the arrow denoting the mental state transition. The derivation rule A.3 is similar to rule A.2 except that there is no substitution θ for which $\varphi\theta'\theta$ is entailed by the agent's effective belief base. As such, an empty substitution is returned and the agent's mental state remains unchanged[2] after executing this step.

A singleton action a is executed when the agent has an intention to do a, and the precondition and context of that action are entailed by the agent's belief base. No variable bindings are returned when a singleton action is executed, and it is followed by the agent immediately executing a rule for the success or failure of that intention depending on whether the agent believes that action to have succeeded or failed. We say that an

[2]More accurately, the agent now believes that it failed to execute this test action and so its belief base is augmented by adding this new fact. However, we ignore this subtlety for the present purpose.

intention *succeeded* when the agent successfully achieves the intended proposition (or gets the intended action done). Therefore, (INTEND self $achieve(p)$) succeeds if the agent believes p after it has persued this intention. On the other hand (INTEND self a) succeeds if (DONE a) is true. Similarly, we say that an intention *failed* if the agent did not achieve its intention. The next two rules denote the computation step one each for success and failure respectively of the intended singleton action.

$$\text{(A.4)}\quad \frac{\begin{array}{l}\langle \text{a,q}\rangle \in \mathcal{I},\ \text{a} \in SAct \\ \mathcal{M}(\text{a}, \langle \mathcal{B}, \mathcal{G}, \mathcal{C}, \mathcal{I}\rangle) = \langle \mathcal{B}', \mathcal{G}, \mathcal{C}', \mathcal{I}'\rangle, \\ \phi = \text{succeded(intend(a,q))} \\ \mathcal{R}(\phi, \langle \mathcal{B}', \mathcal{G}, \mathcal{C}', \mathcal{I}'\rangle) = \langle \mathcal{B}'', \mathcal{G}'', \mathcal{C}'', \mathcal{I}''\rangle\end{array}}{\langle a, \langle \mathcal{B}, \mathcal{G}, \mathcal{C}, \mathcal{I}\rangle\rangle \rightarrow_{\emptyset} \langle nil, \langle \mathcal{B}'', \mathcal{G}'', \mathcal{C}'', \mathcal{I}''\rangle\rangle}$$

$$\text{(A.5)}\quad \frac{\begin{array}{l}\langle \text{a,q}\rangle \in \mathcal{I},\ a \in SAct \\ \phi = \text{failed(intend(a,q))}, \\ \mathcal{R}(\phi, \langle \mathcal{B}, \mathcal{G}, \mathcal{C}, \mathcal{I}\rangle) = \langle \mathcal{B}'', \mathcal{G}'', \mathcal{C}'', \mathcal{I}''\rangle\end{array}}{\langle \text{a}, \langle \mathcal{B}, \mathcal{G}, \mathcal{C}, \mathcal{I}\rangle\rangle \rightarrow_{\emptyset} \langle \text{a}, \langle \mathcal{B}'' \cup \{\phi\}, \mathcal{G}'', \mathcal{C}'', \mathcal{I}''\rangle\rangle}$$

An intention to execute a sequence has two possible computation steps depending on whether or not the first action in the sequence has been done. If the first action has been done then the substitution computed from the execution of the first action of the sequence is applied to the remaining actions and then the remaining actions are intended.

$$\text{(A.6)}\quad \frac{\begin{array}{l}\langle \text{a}_1\text{;a}_2\text{;...;a}_\text{k}\text{,q}\rangle \in \mathcal{I}, \mathcal{B} \nvDash \text{done}(\text{a}_1), \\ \phi = \langle \text{a}_1\text{,intend}(\text{a}_1\text{;a}_2\text{;...;a}_\text{k}\text{,q})\rangle, \phi \notin \mathcal{I}\end{array}}{\langle \mathcal{B}, \mathcal{G}, \mathcal{C}, \mathcal{I}\rangle \rightarrow_{\emptyset} \langle \mathcal{B}, \mathcal{G}, \mathcal{C}, \mathcal{I} \cup \{\phi\}\rangle}$$

$$\text{(A.7)}\quad \frac{\begin{array}{l}\langle \text{a}_1\text{;a}_2\text{;...;a}_\text{k}\text{,q}\rangle \in \mathcal{I}, \mathcal{B} \vDash \text{done}(\text{a}_1\text{;a}_2\text{;...;a}_\text{i}), \\ \langle \text{a}_1\text{;a}_2\text{;...;a}_\text{i}, \langle \mathcal{B},\mathcal{G},\mathcal{C},\mathcal{I}\rangle\rangle \rightarrow_{\theta} \langle nil, \langle \mathcal{B}',\mathcal{G}',\mathcal{C}',\mathcal{I}'\rangle\rangle \\ \phi = \langle \text{a}_{\text{i}+1}\text{;a}_2\text{;...;a}_\text{k}\text{,intend}(\text{a}_1\text{;a}_2\text{;...;a}_\text{k}\text{,q})\rangle, \phi\theta \notin \mathcal{I}\end{array}}{\langle \mathcal{B},\mathcal{G},\mathcal{C},\mathcal{I}\rangle \rightarrow_{\theta'} \langle \mathcal{B}',\mathcal{G}',\mathcal{C}',\mathcal{I}' \cup \{\phi\theta\}\rangle}$$

The computation steps for intention to execute an indefinite repetition a^* is derived by using the transition rules for intention to execute the sequence $a;(a)^*$

The operational semantics of intention to execute a non-deterministic choice of actions is described by two sets of transition rules. The first set of rules specifies the possible computation steps, one for each choice. The second set of rules specifies choosing an

action from the remaining actions upon failure of the chosen action. Here, we specify one rule from each set of transition rules.

$$\text{(A.8)}\quad \frac{\begin{array}{l}\langle \mathrm{a}_1|\mathrm{a}_2|\cdots|\mathrm{a}_\mathrm{k},\mathrm{q}\rangle \in \mathcal{I}\\ \phi_\mathrm{j} = \langle \mathrm{a}_\mathrm{j}, \mathrm{intend}(\mathrm{a}_1|\mathrm{a}_2|\cdots|\mathrm{a}_\mathrm{k},\mathrm{q})\rangle \notin \mathcal{I} \text{ for every } 1 \leqslant \mathrm{j} \leqslant \mathrm{k},\\ \phi_\mathrm{i} = \langle \mathrm{a}_\mathrm{i}, \mathrm{intend}(\mathrm{a}_1|\mathrm{a}_2|\cdots|\mathrm{a}_\mathrm{k},\mathrm{q})\rangle \text{ for some } 1 \leqslant \mathrm{i} \leqslant \mathrm{k},\\ \langle \mathrm{a}_\mathrm{i},\langle \mathcal{B},\mathcal{G},\mathcal{C},\mathcal{I}\cup\{\phi_\mathrm{i}\}\rangle\rangle \rightarrow_\theta \langle nil, \langle \mathcal{B}',\mathcal{G}',\mathcal{C}',\mathcal{I}'\rangle\rangle\end{array}}{\langle \mathcal{B},\mathcal{G},\mathcal{C},\mathcal{I}\rangle \rightarrow_\emptyset \langle \mathcal{B},\mathcal{G},\mathcal{C},\mathcal{I}\cup\{\phi_\mathrm{i}\}\rangle}$$

$$\text{(A.9)}\quad \frac{\begin{array}{l}\langle \mathrm{a}_1|\mathrm{a}_2|\cdots|\mathrm{a}_\mathrm{k},\mathrm{q}\rangle \in \mathcal{I},\ \mathcal{B} \vDash \mathrm{failed}(\mathrm{intend}(\mathrm{a}_\mathrm{m},\mathrm{intend}(\mathrm{a}_1|\mathrm{a}_2|\cdots|\mathrm{a}_\mathrm{k},\mathrm{q})))\\ \phi_\mathrm{j} = \langle \mathrm{a}_\mathrm{j},\mathrm{intend}(\mathrm{a}_1|\mathrm{a}_2|\cdots|\mathrm{a}_\mathrm{k},\mathrm{q})\rangle \notin \mathcal{I} \text{ for every } 1 \leqslant \mathrm{j} \leqslant \mathrm{k}, \mathrm{j} \neq \mathrm{m},\\ \phi_\mathrm{i} = \langle \mathrm{a}_\mathrm{i},\mathrm{intend}(\mathrm{a}_1|\mathrm{a}_2|\cdots|\mathrm{a}_\mathrm{k},\mathrm{q})\rangle \text{ for some } 1 \leqslant \mathrm{i} \leqslant \mathrm{k},\ \mathrm{i} \neq \mathrm{m},\\ \langle \mathrm{a}_\mathrm{i},\langle \mathcal{B},\mathcal{G},\mathcal{C},\mathcal{I}\cup\{\phi_\mathrm{i}\}\rangle\rangle \rightarrow_\theta \langle nil, \langle \mathcal{B}',\mathcal{G}',\mathcal{C}',\mathcal{I}'\rangle\rangle\end{array}}{\langle \mathcal{B},\mathcal{G},\mathcal{C},\mathcal{I}\rangle \rightarrow_\emptyset \langle \mathcal{B},\mathcal{G},\mathcal{C},\mathcal{I}\cup\{\phi_\mathrm{i}\}\rangle}$$

An intention is dropped if it has been found to be impossible to achieve or is irrelevant.

$$\text{(A.10)}\quad \frac{\begin{array}{l}\langle \mathrm{a},\mathrm{q}\rangle \in \mathcal{I}, \mathcal{B}_E \vDash \Box\neg\mathrm{done}(\mathrm{a}) \text{ or } \mathcal{B}_E \vDash \Box\neg \mathit{pre}(\mathrm{a})\\ \text{or } \mathcal{B}_E \vDash \Box\neg \mathit{context}(\mathrm{a}) \text{ or } \mathcal{B}_E \vDash \neg\mathrm{q}\end{array}}{\langle \mathcal{B},\mathcal{G},\mathcal{C},\mathcal{I}\rangle \rightarrow_\emptyset \langle \mathcal{B},\mathcal{G},\mathcal{C},\mathcal{I}\text{-}\{\langle \mathrm{a},\mathrm{q}\rangle\}\rangle}$$

Definition A.15. *Interpreting Internal Commitment*

The goal of a computation step for an internal commitment is to arrive at an intention using one of rational action rules (for example, a rule to intend non-deterministic OR of all actions that can achieve the given commitment).

$$\text{(A.11)}\quad \frac{\begin{array}{l}\langle \varphi,\mathrm{q}\rangle \in \mathcal{C}, \mathcal{B}_E \nvDash \varphi,\ \mathcal{B}_E \nvDash \Box\neg\varphi,\ \mathcal{B}_E \nvDash \neg\mathrm{q},\\ \mathcal{R}(\mathrm{pgoal}(\varphi,\mathrm{q}),\langle \mathcal{B},\mathcal{G},\mathcal{C},\mathcal{I}\rangle) = \langle \mathcal{B}',\mathcal{G}',\mathcal{C}',\mathcal{I}'\rangle\end{array}}{\langle \mathcal{B},\mathcal{G},\mathcal{C},\mathcal{I}\rangle \rightarrow_\emptyset \langle \mathcal{B}',\mathcal{G}',\mathcal{C}',\mathcal{I}'\rangle}$$

An internal commitment is dropped when the committed goal is either believed to be achieved, or impossible to achieve, or is believed to be irrelevant.

$$\text{(A.12)}\quad \frac{\langle \varphi,\mathrm{q}\rangle \in \mathcal{C}, \mathcal{B}_E \vDash \varphi \text{ or } \mathcal{B}_E \vDash \Box\neg\varphi \text{ or } \mathcal{B}_E \vDash \neg\mathrm{q}}{\langle \mathcal{B},\mathcal{G},\mathcal{C},\mathcal{I}\rangle \rightarrow_\emptyset \langle \mathcal{B},\mathcal{G},\mathcal{C}\text{-}\{\langle \varphi,\mathrm{q}\rangle\},\mathcal{I}\rangle}$$

There may also be rules that decide that a commitment is impossible to achieve if it doesn't result in an intention after a certain number of attempts in which case the above transition rule will allow dropping the commitment.

If the commitment is for being in a state where an action expression has been done, that is, commitments of the form $\langle$done(a),q$\rangle$ where $a \in \mathsf{AExp}$ but $a \notin \mathsf{SAct}$ are interpreted using transition rules similar to that for intention to march through the actions in the action expression.

Definition A.16. *Reverse Introspection*

If an agent believes that it has a commitment or an intention but that commitment or intention is not in its commitment or intention base, then it is added to the appropriate mental component. The transition rule for internal commitment is described in the following rule.

$$\frac{\begin{array}{l}\mathcal{B}_E \vDash \text{pgoal(self,p,q)}, \langle\varphi\text{,q}\rangle \notin \mathcal{C}, \\ \mathcal{B}_E \nvDash \text{bel(self,p)} \vee \text{bel(self,}\Box\neg\text{p)} \vee \text{bel(self,}\neg\text{q)}\end{array}}{\langle\mathcal{B},\mathcal{G},\mathcal{C},\mathcal{I}\rangle \rightarrow_{\emptyset} \langle\mathcal{B},\mathcal{G},\mathcal{C} \cup \{\langle\varphi\text{,q}\rangle\},\mathcal{I}\rangle} \tag{A.13}$$

Note that since pgoal(self,p,q) is entailed by the effective belief base, it must be consistent with the existing commitments and intentions of this agent. The transition rule for intention is similar.

A.4 BASIC ACTIONS THEORY

Each STAPLE agent has a number of basic as well as communicative actions in its repertoire. Recall that each singleton action act(t1,..,tn) in STAPLE is defined by the tuple $\langle$x,φ,ψ,σ,$\chi\rangle$ where x is the actor, φ is the precondition, ψ is the post condition, σ is the context, and χ is the "code" of the action. Execution of an action modifies the mental state of an agent. Here, we specify the operational semantics of some of the main basic actions by specifying how they modify the mental state of the agent.

Definition A.17. *Assert, Retract*

The action assert(ϕ) where $\phi \in \mathcal{L}_G$ modifies the belief base of the agent but leaves other parts of the mental state unchanged. Assert reduces the formula ϕ using a set of axioms and the effective belief base (recall that the effective belief base consists of the belief base plus the contents of the commitment stacks), and adds the reduced formulae to the belief base. The consequence of this is that the belief base is in the form as required by the belief reasoner but retains the property that whatever could be inferred from $\mathcal{B} \cup$

$\{\varphi\}$ can also be inferred from $\mathcal{B} \cup \{\psi\}$ where $\psi \in \mathcal{L}_G$ is the reduced form of φ. Let *reduce* be a function of type $\mathcal{L}_G \times \wp(\mathcal{L}_G) \rightarrow \mathcal{L}_G$. The state transformation by assert can then be described in the following rule.

$$\text{(A.14)} \qquad \frac{\phi \in \mathcal{L}_G,\ \mathcal{B}_E \cup \{\phi\} \nvDash \text{false},\ \psi{=}\text{reduce}(\phi, \mathcal{B}_E)}{\langle \text{assert}(\phi),\ \langle \mathcal{B},\mathcal{G},\mathcal{C},\mathcal{I} \rangle \rangle \rightarrow_{\emptyset} \langle nil,\ \langle \mathcal{B} \cup \{\psi\},\mathcal{G},\mathcal{C},\mathcal{I} \rangle \rangle}$$

The axioms used by reduce correspond to the axioms used by the belief reasoner and were discussed in Chapter 4. The action *retract* is the opposite of assert and removes a formula from the belief base.

$$\text{(A.15)} \qquad \frac{\phi\theta \in \mathcal{B}}{\langle \text{retract}(\phi),\ \langle \mathcal{B},\mathcal{G},\mathcal{C},\mathcal{I} \rangle \rangle \rightarrow_{\theta} \langle nil,\ \langle \mathcal{B}\text{-}\{\phi\theta\},\mathcal{G},\mathcal{C},\mathcal{I} \rangle \rangle}$$

Definition A.18. *Adopt, Subgoal, Drop*

The actions adopt, subgoal, and drop correspond to assert and retract except that they act on commitments and intentions instead of the belief base. These actions are typically used from within rules and plans. Adopt adds a new high-level commitment, social commitment, or intention to the agent's mental state. The following rule gives the operational semantics for adopting a new intention.

$$\text{(A.16)} \qquad \frac{\begin{array}{c}\mathcal{B}_E \nvDash \Box\neg \mathit{pre}(\text{a}), \mathcal{B}_E \nvDash \Box\neg \mathit{context}(\text{a}), \nvDash\ \mathcal{B}_E \vDash \neg\text{q}, \\ \mathcal{I} \cup \{\langle \text{a,q} \rangle\} \nvDash \text{false},\ \mathcal{C} \cup \{\text{intend(self,a,q)}\} \nvDash \text{false}\end{array}}{\langle \text{adopt}(\langle \text{a,q} \rangle),\ \langle \mathcal{B},\mathcal{G},\mathcal{C},\mathcal{I} \rangle \rangle \rightarrow_{\emptyset} \langle nil,\ \langle \mathcal{B},\mathcal{G},\mathcal{C},\mathcal{I} \cup \{\langle \text{a,q} \rangle\} \rangle \rangle}$$

The operational semantics for adopting a commitment or social commitment are similar to the above rule and are derived from the definition of those mental attitudes. The basic singleton action *subgoal* is similar to *adopt* except that the new intention, commitment, or social commitment is added to the mental state relative to an existing intention, commitment, or social commitment. *Drop* is the converse of *adopt* and *subgoal* actions. It removes an existing commitment, social commitment, or intention from the agent's mental state.

Next, we modify the language presented in the previous sections to support joint actions, groups, and communicative actions.

A.5 TEAMWORK IN STAPLE

We first modify the language definition and then its operational semantics to support teamwork and communication.

A.5.1 Modifying the Language

Let GVar be the set of group variables having elements (τ,α,..., τ1,τ2 ,..., α1,α2,...). We modify the modal language $\mathcal{L}$ from A.4 by adding formulas for the modal operators mutual belief (mb), and social commitment or persistent weak achievement goal (pwag).

Definition A.19. The Teamwork Language $\mathcal{L}$

1. If $\varphi \in \mathcal{L}$ and $\tau \in$ GVar or $\tau \in$ Agt, then mb(τ,φ) $\in \mathcal{L}$
2. If $\varphi \in \mathcal{L}$ and x,y $\in$ Actor, then mb(x,y,φ) $\in \mathcal{L}$
3. If φ,q $\in \mathcal{L}$ and x,y $\in$ Actor, then pwag(x,y,φ,q) $\in \mathcal{L}$

The mental state of a STAPLE agent now includes a set of social commitments and is denoted by $\langle \mathcal{B},\mathcal{G},\mathcal{C},\mathcal{SC},\mathcal{I} \rangle$ where $\mathcal{SC}$ represents the social commitments of this agent.

Definition A.20. Social Commitments ($\mathcal{SC}$):

An agent's social commitments store $\mathcal{SC}$ consists of tuples $\langle \varphi$,y,q$\rangle$ denoting that this agent is committed towards another agent y for achieving φ relative to some higher-level goal (or context) q. Formally,
$\mathcal{SC} \subseteq \{\langle \varphi$,y,q$\rangle \mid \varphi \in \mathcal{L}$, y $\in$ Agt, q $\in \mathcal{L}\}$ with one of the following constraints:

1. *The agent does not believe φ*
   (a) $\mathcal{B} \not\models \varphi$, that is, φ is not entailed by the belief base.
   (b) $\mathcal{B} \cup \{\neg\varphi\} \not\models$ false, that is, φ is consistent with the belief base.
   (c) $\mathcal{B} \not\models \Box\neg\varphi$, that is, the impossibility of φ must not be entailed by the agent's belief base.
   (d) $\mathcal{B} \not\models \neg$q, that is, the belief base must not entail that the commitment is irrelevant.
   (e) $\mathcal{C} \cup \{\langle \varphi$,q$\rangle\} \not\models$ false, that is, $\mathcal{SC}$ is consistent with $\mathcal{C}$
   (f) $\mathcal{I} \cup \{$pgoal(self,φ,q)$\} \not\models$ false, that is, this social commitment must be consistent with the agent's current intentions.
2. *The agent believes φ*
   Let p = mb(self,y,φ), then

(a) $\mathcal{B} \models \varphi$, that is, φ is entailed by the agent's belief base.

(b) $\mathcal{B} \not\models$ p, that is, p is not entailed by the belief base.

(c) $\mathcal{B} \cup \{\neg p\} \not\models$ false, that is, p is consistent with the belief base.

(d) $\mathcal{B} \not\models \Box\neg$p, that is, the impossibility of p must not be entailed by the agent's belief base.

(e) $\mathcal{B} \not\models \neg$q, that is, the belief base must not entail that the commitment is irrelevant.

(f) $\mathcal{C} \cup \{\langle p,q\rangle\} \not\models$ false, that is, $\mathcal{SC}$ is consistent with $\mathcal{C}$

(g) $\mathcal{I} \cup \{$pgoal(self,p,q)$\} \not\models$ false, that is, this social commitment must be consistent with the agent's current intentions.

3. *The agent believes φ is impossible to achieve*
   Let p = mb(self,y, $\Box\neg\varphi$), then

   (a) $\mathcal{B} \models \Box\neg\varphi$, that is, $\Box\neg\varphi$ is entailed by the agent's belief base.

   (b) $\mathcal{B} \not\models$ p, that is, p is not entailed by the belief base.

   (c) $\mathcal{B} \cup \{\neg p\} \not\models$ false, that is, p is consistent with the belief base.

   (d) $\mathcal{B} \not\models \Box\neg$p, that is, the impossibility of p must not be entailed by the agent's belief base.

   (e) $\mathcal{B} \not\models \neg$q, that is, the belief base must not entail that the commitment is irrelevant.

   (f) $\mathcal{C} \cup \{\langle p,q\rangle\} \not\models$ false, that is, $\mathcal{SC}$ is consistent with $\mathcal{C}$

   (g) $\mathcal{I} \cup \{$pgoal(self,p,q)$\} \not\models$ false, that is, this social commitment must be consistent with the agent's current intentions.

4. *The agent believes that the social commitment is irrelevant*
   Let p = mb(self,y, $\neg$q), then

   (a) $\mathcal{B} \models \neg$q, that is, $\neg$q is entailed by the agent's belief base.

   (b) $\mathcal{B} \not\models$ p, that is, p is not entailed by the belief base.

   (c) $\mathcal{B} \cup \{\neg p\} \not\models$ false, that is, p is consistent with the belief base.

   (d) $\mathcal{B} \not\models \Box\neg$p, that is, the impossibility of p must not be entailed by the agent's belief base.

(e) $\mathcal{C} \cup \{\langle$p,true$\rangle\} \nvDash$ false, that is, $\mathcal{SC}$ is consistent with $\mathcal{C}$

(f) $\mathcal{I} \cup \{$pgoal(self,p,true)$\} \nvDash$ false, that is, this social commitment must be consistent with the agent's current intentions.

Definition A.21. *Effect of social commitment on other mental state components*

The internal commitments and intentions of an agent must be consistent with its social commitments. As such, we add the following constraints to A.7 of these mental state components.

- $\mathcal{SC} \cup \{$pgoal(self,φ,q)$\} \nvDash$ false, that is, this commitment must be consistent with the agent's social commitments.
- $\mathcal{SC} \cup \{$intend(self,a,q)$\} \nvDash$ false, that is, this intention must be consistent with the agent's social commitments.

After including social commitment to the mental state of an agent, the type of mental state $\mathcal{T}_M$is now given by $\wp(\mathcal{L}_G) \times \wp(\mathcal{L}_G) \times \wp(\mathcal{L}_G \times \mathcal{L}_G) \times \wp(\mathcal{L}_G \times$ Agt$\times \mathcal{L}_G) \times \wp($AExp$\times \mathcal{L}_G)$.

Definition A.22. *Introspection*

STAPLE agents are introspective with respect to their social commitments (along with internal commitments and intentions). Therefore, the definition of effective belief base (A.8) is modified to include the social commitments of the agent. Formally, we add the following statement to that definition:

- if $\langle\varphi$,y,q$\rangle \in \mathcal{SC}$, then pwag(self, y,φ,q) $\in \mathcal{B}_E$

Next we discuss the effects on operational semantics of STAPLE due to introduction of the social commitment.

A.5.2 Modifying the Operational Semantics

The reference to mental state in all the transition rules for the operational semantics listed in Section A.3 must be modified to include the social commitment, that is, they must be replaced by $\langle\mathcal{B},\mathcal{G},\mathcal{C},\mathcal{SC},\mathcal{I}\rangle$. Moreover, the operational semantics in Section A.3 must be augmented to include the transition rules for interpreting social commitments.

Definition A.23. *Interpreting Social Commitments*

Social commitments always lead to internal commitments. The following four transition rules specify the computation steps to arrive at an internal commitment from a social commitment. The next rule says to add an internal commitment if the agent does not believe the socially committed goal, and if it is reasonable to add that commitment.

$$\text{(A.17)} \qquad \frac{\langle\varphi,\text{y},\text{q}\rangle \in \mathcal{SC},\ \mathcal{B}_E \not\models \varphi,\ \mathcal{B}_E \not\models \Box\neg\varphi,\ \mathcal{B}_E \not\models \neg\text{q},\ \mathcal{C} \not\models \langle\varphi,\text{q}\rangle}{\langle\mathcal{B},\mathcal{G},\mathcal{C},\mathcal{SC},\mathcal{I}\rangle \rightarrow_{\emptyset} \langle\mathcal{B},\mathcal{G},\mathcal{C} \cup \{\langle\varphi,\text{q}\rangle\},\mathcal{SC},\mathcal{I}\rangle}$$

The next three rules require adding an internal commitment to establish mutual belief with the agent towards whom this agent is socially committed about the achievement, impossibility, or irrelevance of the socially committed goal.

$$\text{(A.18)} \qquad \frac{\begin{array}{c}\langle\varphi,\text{y},\text{q}\rangle \in \mathcal{SC},\ \mathcal{B}_E \models \varphi,\ \text{p=mb(self,y,}\varphi\text{)},\\ \mathcal{B}_E \not\models \text{p},\ \mathcal{B}_E \not\models \Box\neg\text{p},\ \mathcal{B}_E \not\models \neg\text{q},\ \mathcal{C} \not\models \langle\text{p,q}\rangle\end{array}}{\langle\mathcal{B},\mathcal{G},\mathcal{C},\mathcal{SC},\mathcal{I}\rangle \rightarrow_{\emptyset} \langle\mathcal{B},\mathcal{G},\mathcal{C} \cup \{\langle\text{p,q}\rangle\},\mathcal{SC},\mathcal{I}\rangle}$$

$$\text{(A.19)} \qquad \frac{\begin{array}{c}\langle\varphi,\text{y},\text{q}\rangle \in \mathcal{SC},\ \mathcal{B}_E \models \Box\neg\varphi,\ \text{p=mb(self,y,}\Box\neg\varphi\text{)},\\ \mathcal{B}_E \not\models \text{p},\ \mathcal{B}_E \not\models \Box\neg\text{p},\ \mathcal{B}_E \not\models \neg\text{q},\ \mathcal{C} \not\models \langle\text{p,q}\rangle\end{array}}{\langle\mathcal{B},\mathcal{G},\mathcal{C},\mathcal{SC},\mathcal{I}\rangle \rightarrow_{\emptyset} \langle\mathcal{B},\mathcal{G},\mathcal{C} \cup \{\langle\text{p,q}\rangle\},\mathcal{SC},\mathcal{I}\rangle}$$

$$\text{(A.20)} \qquad \frac{\begin{array}{c}\langle\varphi,\text{y},\text{q}\rangle \in \mathcal{SC},\ \mathcal{B}_E \models \neg\text{q},\ \text{p=mb(self,y,}\neg\text{q)},\\ \mathcal{B}_E \not\models \text{p},\ \mathcal{B}_E \not\models \Box\neg\text{p},\ \mathcal{C} \not\models \langle\text{p,true}\rangle\end{array}}{\langle\mathcal{B},\mathcal{G},\mathcal{C},\mathcal{SC},\mathcal{I}\rangle \rightarrow_{\emptyset} \langle\mathcal{B},\mathcal{G},\mathcal{C} \cup \{\langle\text{p,true}\rangle\},\mathcal{SC},\mathcal{I}\rangle}$$

Definition A.24. *Reverse Introspection*

STAPLE agents are also capable of reverse introspection (A.16) with respect to their social commitments. If an agent's belief that it has a social commitment is not in its social commitment store, then it is added to the appropriate mental component. The transition rule for social commitment is described in the following rule.

$$\text{(A.21)} \qquad \frac{\begin{array}{c}\mathcal{B}_E \models \text{pwag(self,y,p,q)},\ \langle\varphi,\text{y},\text{q}\rangle \notin \mathcal{SC},\\ \mathcal{B}_E \not\models \text{mb(self,y,p)} \vee \text{mb(self,y,}\Box\neg\text{p)} \vee \text{mb(self,y,}\neg\text{q)}\end{array}}{\langle\mathcal{B},\mathcal{G},\mathcal{C},\mathcal{SC},\mathcal{I}\rangle \rightarrow_{\emptyset} \langle\mathcal{B},\mathcal{G},\mathcal{C},\mathcal{SC} \cup \{\langle\varphi,\text{y},\text{q}\rangle\},\mathcal{I}\rangle}$$

Note that since pwag(self,y,p,q) is entailed by the effective belief base, it must be consistent with the existing commitments, social commitments, and intentions of this agent. Also, when mb is entailed by the belief base, it effectively represents a bmb.

We now discuss changes to the action theory as a result of introducing social commitments.

A.5.3 Modifying the Action Theory

The reference to mental state in all the transition rules for the basic actions theory in Section A.3 must be modified to include the social commitment, that is, they must be replaced by $\langle \mathcal{B},\mathcal{G},\mathcal{C},\mathcal{SC},\mathcal{I}\rangle$.

The support for multiple agents and the social commitment of one agent directed towards another requires that agents be able to communicate with each other. Communicative acts are similar to the basic actions and they affect the mental state of the communicating agents (including that of over-hearers). The transition rules for the main communicative actions in STAPLE for communication between two agents are defined next.

Definition A.25. *Communicative Actions*

The operational semantics of the communicative actions *request*, *inform*, *agree*, and *refuse* is derived from their logical definitions in [106, 65]. The next transition step specifies change of mental state for both the actor (sender) and the recipient of the request action.

$$\frac{\begin{array}{c}\psi = \text{done(y,a)} \wedge \text{pwag(y,x,done(y,a),pwag(x,y,done(y,a),q)}\wedge\text{q)} \\ \varphi = \text{pwag(x,y,}\psi\text{,q)}, \mathcal{B}_E \cup \{\text{mb(x,y,}\varphi\text{)}\} \nvDash \text{false}\end{array}}{\langle \text{request(x,y,a,q)}, \langle \mathcal{B},\mathcal{G},\mathcal{C},\mathcal{SC},\mathcal{I}\rangle\rangle \rightarrow_{\emptyset} \langle \mathit{nil}, \langle \mathcal{B} \cup \{\text{mb(x,y,}\varphi\text{)}\},\mathcal{G},\mathcal{C},\mathcal{SC},\mathcal{I}\rangle\rangle} \tag{A.22}$$

Note that the reverse introspection step will result in the sender's adopting a social commitment towards the recipient if it does not yet have that social commitment. The inform action establishes mutual belief that the sender believes the informed proposition (assuming that nothing has changed since the inform was performed, and the act of informing did not change that proposition).

$$\frac{\varphi = \text{bel(x,p)},\ \mathcal{B}_E \cup \{\text{mb(x,y,}\varphi\text{)}\} \nvDash \text{false}}{\langle \text{inform(x,y,p)}, \langle \mathcal{B},\mathcal{G},\mathcal{C},\mathcal{SC},\mathcal{I}\rangle\rangle \rightarrow_{\emptyset} \langle \mathit{nil}, \langle \mathcal{B} \cup \{\text{mb(x,y,}\varphi\text{)}\},\mathcal{G},\mathcal{C},\mathcal{SC},\mathcal{I}\rangle\rangle} \tag{A.23}$$

Note (for later use) that in the complete action theory of inform, we would also specify the enabling invariant (context) for performing an *inform* that the sender x must believe the proposition being informed. The transition rules for agree is similar to that of inform except for a different proposition φ.

$$\frac{\begin{array}{c}\varphi\text{= pwag(x,y,done(x,a),pwag(y,x,done(x,a),q)}\wedge\text{q)}, \\ \mathcal{B}_E \cup \{\text{mb(x,y,}\varphi\text{)}\} \nvDash \text{false}\end{array}}{\langle \text{agree(x,y,a,q)}, \langle \mathcal{B},\mathcal{G},\mathcal{C},\mathcal{SC},\mathcal{I}\rangle\rangle \rightarrow_{\emptyset} \langle \mathit{nil}, \langle \mathcal{B} \cup \{\text{mb(x,y,}\varphi\text{)}\},\mathcal{G},\mathcal{C},\mathcal{SC},\mathcal{I}\rangle\rangle} \tag{A.24}$$

Next, we discuss the support in STAPLE for convenience constructs such as plans.

A.6 REAL WORLD OPTIMIZATIONS

Plans are well known construct in agent programming. A plan consists of an action expression that is designed to achieve a particular goal. Similar to actions, plans have an effect, a precondition, and a context. However, instead of "code", STAPLE plans have a body consisting of either singleton actions or other plans.

Definition A.26. *Plan*

A STAPLE plan is a named action expression. In general, the precondition, context, etc. for a plan can be derived from the corresponding formulae of the constituent actions. However, we require these terms to be explicitly specified. A STAPLE plan act(t1,..,tn) is defined by a tuple $\langle\varphi,\psi,\sigma,\beta\rangle$
where, act $\in$ ActSym is the name of the plan,

t1,...,tn $\in$ Term are the parameters to the plan,

$\varphi,\psi,\sigma \in \mathcal{L}$, and specify the precondition, the post-condition, and the context of the plan

$\beta \in$ AExp and $\beta \notin$ ActionVar is the plan body

Note that the constituent actions of a plan may define different agents as actors and therefore, a plan may have multiple actors in which case it is considered a joint action.

Recall that STAPLE rules for rational action (A.10) have a premise and a consequent that is a basic action (or action expression). Also, recall that the semantics of rule execution (A.13) uses one of the conjuncts in the premise as an applicability criterion. For reasons of efficiency, STAPLE requires explicit specification of the applicability condition of a rule. STAPLE interpreters can then use the applicability criterion to quickly narrow down the set of rules applicable to a given situation. As such, we modify rule A.10 as follows.

Definition A.27. *Rules for Rational Action*

Rules in a STAPLE program have three components: an applicability condition, the premise, and the action to execute if the premise is true. As such rules can be thought of as a conditional action and are represented as

$\langle\varphi, \psi\rangle \rightarrow a$

where, $\varphi \in \mathcal{L}$ is the applicability condition,

$\psi \in \mathcal{L}$ is the premise of the rule, and

$a \in$ AExp is an expression such that every component of a is a basic action.

STAPLE defines a primitive action to execute rules whose semantics remains unchanged from that specified in A.13.

In the real world, agents may have multiple commitments and intentions competing for resources and attention. Furthermore, several rules may be applicable to a given situation and multiple actions and plans that can be achieve a given goal. One way to prioritize these competing means and ends is to use an importance function.

Definition A.28. *Importance*

The rules, plans, commitments, social commitments, and intentions have an importance rating that is used to prioritize them when needed. So we modify the definition of the relevant language constructs we have seen so far to add a function that returns the importance. For simplicity (and what is actually implemented in the interpreter discussed in Chapter 4), let it be the constant function n for now. The definitions of the following language constructs get modified as specified below:

- Plan: $\langle\varphi,\psi,\sigma,\beta,\text{n}\rangle$
- Rule: $\langle\varphi,\ \psi,\ \text{n}\rangle \rightarrow a$
- Commitment: $\langle\text{p,q,n}\rangle$
- Social Commitment: $\langle\text{p,y,q,n}\rangle$
- Intention: $\langle\text{a,q,n}\rangle$

One impact of the notion of importance is that all references to plans, rules, commitments, intentions, and social commitments in the operational semantics and elsewhere are modified as per the A.28. Its most pronounced impact is on the operational semantics of intention to execute an action.

Definition A.29. *Intention to execute a singleton action*

A singleton action a is executed when the agent has an intention to do a, the precondition and context of that action are entailed by the agent's belief base, and this *intention happens to be one of the most important intentions of the agent.* The next two rules denote the computation step one each for success and failure respectively of the intended

singleton action and are modified from rules A.4 and A.5.

$$\frac{\begin{array}{l}\langle \text{a,q,n}\rangle \in \mathcal{I},\ \text{a} \in SAct, n = \max\ i\ \text{s.t.}\langle \text{a,q,}i\rangle \in \mathcal{I},\\ \mathcal{M}(\text{a,}\langle \mathcal{B},\mathcal{G},\mathcal{C},\mathcal{SC},\mathcal{I}\rangle) = \langle \mathcal{B}',\mathcal{G},\mathcal{C}',\mathcal{SC}',\mathcal{I}'\rangle,\\ \phi = \text{succeded(intend(a,q,n))}\\ \mathcal{R}(\phi,\langle \mathcal{B}',\mathcal{G},\mathcal{C}',\mathcal{SC}',\mathcal{I}'\rangle) = \langle \mathcal{B}'',\mathcal{G}'',\mathcal{C}'',\mathcal{SC}'',\mathcal{I}''\rangle\end{array}}{\langle a, \langle \mathcal{B},\mathcal{G},\mathcal{C},\mathcal{SC},\mathcal{I}\rangle\rangle \rightarrow_{\emptyset} \langle nil, \langle \mathcal{B}'',\mathcal{G}'',\mathcal{C}'',\mathcal{SC}'',\mathcal{I}''\rangle\rangle} \tag{A.25}$$

$$\frac{\begin{array}{l}\langle \text{a,q,n}\rangle \in \mathcal{I}, a \in SAct, n = \max\ i\text{s.t.}\langle \text{a,q,}i\rangle \in \mathcal{I},\\ \phi = \text{failed(intend(a,q,n))},\\ \mathcal{R}(\phi,\langle \mathcal{B},\mathcal{G},\mathcal{C},\mathcal{SC},\mathcal{I}\rangle) = \langle \mathcal{B}'',\mathcal{G}'',\mathcal{C}'',\mathcal{SC}'',\mathcal{I}''\rangle\end{array}}{\langle \text{a,}\langle \mathcal{B},\mathcal{G},\mathcal{C},\mathcal{SC},\mathcal{I}\rangle\rangle \rightarrow_{\emptyset} \langle \text{a,}\langle \mathcal{B}'' \cup \{\phi\},\mathcal{G}'',\mathcal{C}'',\mathcal{SC}'',\mathcal{I}''\rangle\rangle} \tag{A.26}$$

Next, we conclude with a discussion on the relationship between the operational semantics and JI theory.

A.7 DISCUSSION

STAPLE and JI theory are connected in at least two different ways. First, the joint intention theory provides the axiomatic/logical semantics of STAPLE. As such, the formulas in the agent's belief base have Kripke's weak S5 semantics and those in its goal base have system K semantics. Further, the formulas in the goal base are constrained to be compatible with the agent's belief base. The logical semantics of internal commitment is given by PGOAL, that of intention by INTEND, and that of social commitment by PWAG. The entailment relationship used in the previous sections comes from this JI based logical semantics. In fact, the axioms describing the logical semantics of each modality and their relationship with each other are needed by the belief reasoner behind implementations of STAPLE.

Second, the model theory for the logic of joint intentions provides the denotational semantics for the corresponding terms in STAPLE. As such, the terms in STAPLE have the same possible world semantics as the corresponding terms in its logical semantics.

The communicative acts within the framework of JI theory are defined as attempts having an associated goal and an intention. As such, the transition rules A.22 to A.24 are derived from the intention part of the logical semantics of the corresponding communicative acts assuming that the intent behind the communication was successful. The definitions of intention, commitment and social commitment in this appendix follow directly from the definitions of INTEND, PGOAL, and PWAG in the JI logic. Similarly, the

derivation rules for intending various action expressions (Definition A.14) and for commitment for getting different action expressions done (Definition A.15) correspond directly to results in the logic (Table 2.1 and Table 2.2).

The language definition and its operational semantics presented in this appendix operationalize the definitions of modal constructs such as PGOAL in a way that can be directly implemented in software. Recall that the semantics of the modal concepts PGOAL, INTEND, and PWAG is built into the interpreter by using the logical definition of these terms. For instance, the definition of PGOAL in the logic says that an agent having a PGOAL that p relative to a higher level goal q does not believe p, and has a goal that eventually p, and it must keep this goal at least until it believes that p has been achieved or is impossible to achieve or is irrelevant. The operational semantics of PGOAL on the the other hand goes into explicit details about how the various belief and commitment data structures are updated, what consistency checks must be done, etc. that is needed for encoding support for PGOAL into a JI interpreter.

Biographical Note

Sanjeev Kumar was born in August, 1974 in Patna, India. He received his B.Tech. degree in Electrical Engineering from the Indian Institute of Technology, Kanpur, India in 1995 and his M.S. degree in Computer Science from the Oregon Health & Science University, USA in 2002. Thereafter, he worked as a staff scientist for three years at the Center for Human-Computer Communication in the Department of Computer Science and Engineering at the Oregon Health & Science University. Mr. Kumar has received several scholarships including the academic proficiency prize for the best undergraduate project at the Indian Institute of Technology, Kanpur, India. His research interests include natural language dialogue, multiagent systems, theories of collaboration and communication, and multimodal interaction. He has published more than a dozen scientific papers in peer-reviewed international conferences and journals and has more than 25 patents pending at the US Patent and Trademark Office.