# A Logic-Based Model of Intention Formation and Action for Multi-Agent Subcontracting*

John Grant
Dept. of Computer and Information Sciences
and Department of Mathematics
Towson Univ.
Towson, MD 21252 USA
jgrant@towson.edu

Sarit Kraus
Dept. of Computer Science
Bar-Ilan Univ., Ramat-Gan, 52900 Israel
and Institute for Advanced Computer Studies
Univ. of Maryland, College Park, MD 20742
sarit@umiacs.umd.edu

Donald Perlis
Dept. of Computer Science
and Institute for Advanced Computer Studies
Univ. of Maryland,
College Park 20742, USA
perlis@cs.umd.edu

**Abstract**

We present a formalism for representing the formation of intentions by agents engaged in cooperative activity. We use a syntactic approach presenting a formal logical calculus that can be regarded as a meta-logic that describes the reasoning and activities of the agents. Our central focus is on the evolving intentions of agents over time, and the conditions under which an agent can adopt and maintain an intention. In particular, the reasoning time and the time taken to subcontract are modeled explicitly in the logic. We axiomatize the concept of agent interactions in the meta-language, show that the meta-theory is consistent and describe the unique

*This research was supported by NSF under grant IIS-0222914 and by the Air Force Office of Scientific Research, and the Office of Naval Research. Preliminary version appeared in the proceedings of AAAI-2002. We wish to thank the referees for many helpful comments and suggestions.

intended model of the meta-theory. In this context we deal both with subcontracting between agents and the presence of multiple recipes, that is, multiple ways of accomplishing tasks. We show that under various initial conditions and known facts about agent beliefs and abilities, the meta-theory representation yields good results.

**Keywords:** intentions, subcontracting, cooperative agents, syntactic logic, minimal model semantic.

# 1 Introduction

In this paper we provide a formal meta-theory of the formation of intentions among multiple cooperating agents.[1] Such a formal theory is very useful to properly understand computational models of such cooperative behavior, particularly as a specification language. The meta-logic can be used to analyze the behavior of a team of agents and prove various theorems about their activities. We provide a single unified framework that captures with a relatively small number of predicates and axioms an evolving notion of time, the formation of potential and actual intention of agents, subcontracting between agents, actions involving a recipe tree of subactions, parallel actions, multiple recipes for actions, and the performance or failure of agents doing these actions.

For instance, agent 5 may know that in order to perform a complex action *a* it is sufficient to perform subaction *b* and subaction *c*. However, agent 5 cannot do *c* but agent 3 can do *c*. If agent 5 is asked to perform action *a* it can subcontract to agent 3 the performance of action *c*. There is a rule here that the agent follows: when it cannot perform an action (or subaction), it finds an agent that can do the action and subcontracts the action to it. In this paper we specify such rules as meta-axioms relating to the mental states of agents in the sense of the intentions that the agents adopt to perform certain actions and how these lead to the actual performance of the actions. If the agents have been designed to always apply these rules in a cooperative manner, then we can prove what actions will actually be performed.

We use the approach of the mental state model of plans [42]. Pollack's definition of the individual plan of an individual agent to do an action $\alpha$ includes four constituent mental attitudes: (1) belief that performance of certain actions $\beta_i$ would entail performance of $\alpha$; the $\beta_i$ constitute "a recipe for $\alpha$;" (2) belief that the agent could perform each of the $\beta_i$; (3) intentions to do each of the $\beta_i$; (4) an intention to do $\alpha$ by doing the $\beta_i$. This definition has been extended to handle multi-agent systems in various ways (e.g., [21, 19, 20]). Thus, in this approach, agents do not model explicitly the state of the world. In contrast to classical planning there is no attempt to search for a sequence of actions that change the world state from its initial state to the goal state and thus the focus is not on the problems associated with modeling how the state of the world is changing over time [47]. Instead, an agent attempts to perform actions and in its planning it decomposes complex actions to simpler actions using a library of recipes. The agent maintains a belief set that includes beliefs of what it can and can't do at various times which are used to decide on whether to adopt intentions. These beliefs may

[1] Our focus is on how intentions are formed and action gets done in this setting. For detailed discussions of intentions and actions see [35, 41, 11].

change over time based on observations and reasoning which may lead to dropping intentions and adopting new ones.

Most of the previous work on the mental state model for planning lack a well established semantics, and especially lack a formal treatment of time. In a previous work [18] we made the first step in providing semantics for this approach by presenting a framework for the way the beliefs of bounded rational agents change over time. In this paper we consider intention formation when agents can subcontract actions to one another. We focus on how the intentions of the agents change over time.

We assume all agents to be cooperative; but giving this requirement a formal characterization involves the notion of intention. We require an agent to adopt a *potential intention* to perform an action if it is asked to do so. We provide a specification for the beliefs and intentions that an agent should have in order to turn a potential intention to an intention. We also specify the communication needed and the change in the agents' beliefs and intentions when something goes wrong.

Our work differs from others in its use of a meta-theory of intentions; of an evolving notion of time that allows agents to reason about genuinely-approaching deadlines; and of a reified notion of (explicit) agent beliefs so that they can be tracked over time. Thus we view the reasoning of agents as ongoing processes rather than as fixed sets of conclusions. This reasoning involves rapidly evolving sets of beliefs and intentions, and is governed in part by formal rules of inference. We model beliefs and intentions in a formal logical agent calculus that can be regarded as a meta-logic describing an actual on-board agent logic that evolves in (and takes account of) that same passage of time. This is a style of formal representation that departs from traditional temporal logics in that there is an internally evolving notion of time that the logic must keep track of.

For allowing this kind of reasoning, our approach utilizes a strongly sorted calculus, distinguishing the application language, time, and various syntactic sorts. We provide formal tools for representing agent intentions and methods for reasoning with and about such intentions, especially in the context of cooperative actions. We focus on cases where one agent alone cannot accomplish a complex task and must subcontract with other agents.

The format of this paper is as follows. In the rest of Section 1 we describe the basic syntactic formalism for intentions and give an example that is used throughout the paper for illustration. Section 2 deals in more detail with the syntax and semantics of the meta-language. In Section 3 we show how our framework can be used to model the case of cooperative multiple agents where each action has a single recipe. Multiple recipes for actions are considered in Section 4. Some work related to this paper is described in Section 5. Section 6 summarizes the paper and indicates topics for future research. We also provide four tables: the first one summarizes the convention used for variables in Section 2.1; the others at the end of the paper give the meanings of the predicates for two different sections and specify the locations of the axioms.

## 1.1 Languages for Reasoning About Agents

In our framework we find it useful to introduce three related languages. The first language, $L_S$, is the language of the subject matter, that is, the subject that the agents

reason about: e.g. web searches. The second language, $L_A$, is the "agent" language. This includes the subject matter language, but also has additional symbols allowing assertions about what some particular agent may believe at a given moment; this allows an agent to reason about another agent, for instance in deciding whether to ask another agent to perform some subtask based on what that other agent believes.[2] The third language, $L_M$, is the meta-language used to reason about the agents.

The meta-language, $L_M$, must be powerful enough to characterize agent behavior quite generally. For instance, in our meta-language we can write the statement that an agent always applies Modus Ponens, when that is possible. This, taken as an axiom, would assert that the agents are programmed to apply Modus Ponens; that is an aspect of their behavior. Thus, while the agent language, $L_A$, must allow for the expression of the formulas *A*, *A implies B*, and *B*, still an agent does not in general have the meta-rule such as "I apply Modus Ponens whenever I can," even if that happens to be true (and expressed in the meta-language). The three languages are sorted first-order logics.

In much of the existing formal work on modeling agent behavior, it is not specified precisely *who* is doing the reasoning; it appears implicit that the reasoning is meta-reasoning by external observers about agent behavior. In particular, we use our meta-language to track the evolving behavior (including the evolving beliefs and intentions) of the agents over time. Hence the meta-language (as well as the agent language) must have a robust notion of time. It is an important aspect of this approach that we explicitly represent individual beliefs of an agent, i.e., we apply a syntactic approach where beliefs are objects (such as sentences) rather than a modal approach with abstract (modal) propositions. For instance, an agent at time t may believe $A$, $A \rightarrow B$, and $B \rightarrow C$, and yet not believe $C$, simply because it did not yet (have time to) apply Modus Ponens (twice); and yet it may at a later time believe $C$ (e.g., at time t+2, if it applies MP once at each time step). In order to represent this sort of behavior we need names for beliefs; we also need names for other sentences, such as statements about agent intentions. For instance, agents invoke a Tell process in order to inform one another of a new intention, as part of a cooperative venture to achieve some goal. We write roughly in the form Tell("Int(...)"); so Tell takes an argument that is the name of the sentence Int(...) that expresses an intention.

The syntactical approach that we follow seems attractive to researchers in artificial intelligence (see chapters 2 and 8 of [12]). For one thing, it is natural to represent the knowledge and beliefs of a computer program by writing sentences representing facts that are known or believed. For another, syntactical treatments can be formalized in the classical predicate calculus, which is the *lingua franca* of knowledge representation [22, 36, 40]; since there is good theorem-proving technology for this language (e.g. [3, 4, 6]), such treatments lend themselves directly to implementations.

Furthermore, in other approaches to the issue of agent reasoning, often logical omniscience is assumed. This means that the agent's beliefs are closed under logical rules, such as Modus Ponens, in every time period. That is, if the formulas $A$ and $A \rightarrow B$ are believed by the agent at time $t$, the agent is assumed to believe $B$ at the *same* time. We think that our approach, where rules are applied over time is a more realistic

[2] We do not assume that the agent uses logic for their reasoning. $L_A$ is the language that is used to specify the agent behavior. We do assume that each agent maintains a database with – possibly fallible – information about agent abilities. However, we do not make any assumptions about the specification of the database.

way to model agent beliefs and intentions. Some researchers avoided the use of the syntactic approach because, in general, the syntactical treatments face a serious theoretical difficulty [38, 13]. Richard Montague [34] showed that under certain fairly intuitive conditions a syntactic formalization of modal notions such as knowledge becomes contradictory. Thomason later showed [53] that under even less restrictive conditions a syntactic formalization yields a contradictory set of beliefs. However, we showed in [18] that when dealing with only beliefs our meta-theory is always consistent. Moreover an agent theory need not be inconsistent either. Nevertheless, it is an interesting property of our approach that agent theories may indeed be (or become) inconsistent — e.g., through observations — and this is one of the advantages of our treatment: commonsense reasoning can proceed in the presence of inconsistency. The time-situated character of our logic offers "protection" from inconsistency in two ways: it affords a consistent meta-theory, and it allows for an inconsistent but still useful agent theory. We further extends these results for intentions as discussed in section 2.2 below.

Finally, we note here that we regard actions as objects. Instead of "make [arrive-on-time] true" our actions have the flavor "arrive on time". This seeming mere terminologic difference has the implication that actions can be treated on a par with names of beliefs, making the formal treatment more uniform.

## 1.2 Agent Beliefs and Intentions

Beliefs and intentions are the two main predicates in our logic. In [18] we focus on the way the beliefs of bounded rational agents change over time. In this paper we focus on the intention formation of cooperative bounded agents and discuss the agents' beliefs only when they are necessary for understanding the agents' intention formation process. We first survey the way agents' beliefs are changing over time. The specific axioms are presented in [18]. In our context, these beliefs could be on the actions that an agent can do, and the actions that other agents can perform. Changes in such beliefs will lead to changes in intentions.

Because of our emphasis on changing beliefs and intentions over time, agents need to know the present time and be able to reason about time. In addition, the meta-language should be able to reason about time and about the change of the agents' beliefs and intentions over time. In both logics, we use a discrete model of time.

Also, in our framework agents have introspection capabilities [18]. By this we mean that one of the agent's beliefs may be that it believed or did not believe something at a different time. There are two types of introspection: positive and negative. An example of positive introspection is if an agent believed some fact (represented by a formula) at time $t$, it will then believe at time $t + 1$ that it believed that fact at time $t$. This way an agent can reason about its own or another agent's beliefs over time. Suppose now that at time $t$ the agent is considering but does not believe a particular possible fact $A$ (that is, $A$ is not in its database). Then at the next time value, $t + 1$, it will believe (know) by negative introspection that it did not believe $A$ at time $t$.

Another important capability of an agent is the inheritance of beliefs, or as it is sometimes called, persistence [49, 37]. The idea is that if an agent believes some fact $A$ and does not believe something contradictory to it (that is, $not\ A$), it will continue to

believe $A$. There are exceptions to inheritance. Things that are changing, such as the location of a moving object, or the present time value, should not be inherited.

We have found it useful also to add axioms concerning the cooperation of agents. So for example if an agent "tells" another agent a fact, the second agent will treat it as a fact. This assumes that agents are trustworthy. Agents may also be helpful to other agents by answering each other's questions. Note that of course there can be nontrustworthy agents, and that this is a major topic of research (see a survey in [27]). But here we focus on fully cooperative (and nonforgetful) agents.

The exact definition of intentions may be necessary for cooperation. For example, an agent needs to know how to interpret a belief that its partner has adopted an intention to do the given action: does this mean the action will indeed be performed by that partner if the partner is able? Thus, it is important to formally define (a) what are the minimal requirements that are needed for an agent to adopt an intention; (b) how an agent's intentions change over time (and what are the occasions when an agent will drop an intention); (c) what are the reasoning processes that the agent will perform when it adopts an intention; and (d) what are the actions that an agent will perform when it adopts an intention.

We define a notion of "potential" intention that means roughly that the agent has been given a task and is determining whether it can do it (perhaps with help). We reserve "intention" for the situation in which the agent has determined that it can do the task (perhaps with help) and in that case it will indeed attempt to carry out the task, unless in the meantime it has dropped that intention because (a) it is told to drop it; or (b) it is given another task that conflicts with the first one and which is assigned higher preference; or (c) the world changes and it finds that it can no longer do the task (perhaps because a subcontracting agent cannot do its part). The main difference between intentions and desires [14, 28] is the strong commitment to perform the intended actions. While such a commitment is not associated with a potential intention, the agent that has a potential intention to do an action is committed to consider the subactions (if any) required for the action (in a recipe) and possibly communicate with other agents to get their commitments for all the required subactions.

In the context of collaboration, it is also important to decide whether an agent can have intentions that another agent will perform an action. In our model, an agent can't intend directly that another agent will do an action[3]; however, its plan for doing $a$, that is motivated by the intention to do $a$, may include the intentions of other agents.

We divide actions into two types: basic level and complex. Basic level actions are done by agents that can do them as primitive acts. A complex action requires a *recipe* [42], that is, a specification of a list of actions, the doing of which constitutes performance of the complex action[4]. The subsidiary actions, referred to as subactions, may be basic level or complex. The general situation is illustrated in the tree of Figure 1 where we consider only a single recipe for a complex action. Here an agent is given task $a1$, which has a recipe shown in the tree, i.e. to do $a1$, it is sufficient to do $b1$,$b2$,$b3$

[3]Technically, intending that another agent do an action is impossible in our logic since done and intentions are sentences and the intentions are with respect to actions not sentences. Furthermore, since agents are autonomous, it is not reasonable to assume that one agent intends that another agent will do an action [33].

[4]We assume that for each subaction, its relative start and maximal possible end time are specified in the recipe.

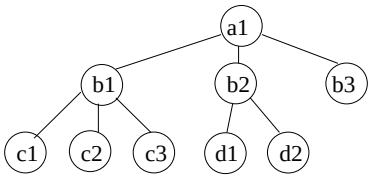


Figure 1: An example of a recipe tree.

(in that order). Similarly, $b1$ and $b2$ themselves have recipes. The leaves are the basic level actions. If an agent cannot do all the actions for a recipe, it then may subcontract some of them to other agents.

## 1.3 Example

We now illustrate many of our ideas with an example that will be used in the rest of the paper. We have chosen this example to be relatively simple and yet to have a number of the more interesting features that our work allows. We assume that there are two agents, one of which has been given a request to perform an action starting at a particular time. We further assume for now that for each action (and subaction) there is only one available recipe for performing that action. We use the convention that an agent must always subcontract a complex action if it cannot do the first subaction. We may imagine that the agents are part of a network of helper agents that work in the service of a human.

Let the two agents be G and H. The request to act is given to G, and the request itself is (for G) to arrange (for the human requester) to get to the airport on time. Let us name the action "aaat" (arrange for arriving at airport on time).

There is a recipe for aaat, consisting of four subactions:

**(i)** find flight info (number, date, airport)

**(ii)** use this info to find time to arrive at airport

**(iii)** estimate time to leave for airport

**(iv)** arrange for a taxi

Furthermore, the recipe for action (ii) consists of two subactions: (a) find the url for the airline and (b) do a web search; while the recipe for action (iv) consists of the two subactions: (c) find the phone number for the taxi and (d) call for the taxi. We suppose that G can carry out by itself (i), (ii)(a), (iii), and (iv)(d), but it needs to subcontract (ii)(b) and (iv) to H which, not being able to do (iv)(d) will subcontract this to G. Of course, G and H have to be especially designed to be able to do (or not do ) just the right things for this example to work. In a more realistic (but far more complex) example, there would be more agents, and subcontracting in general would involve an agent asking one agent after another to help with a subaction until success occurred, and even then the success might later be overturned due to some later failure. Some

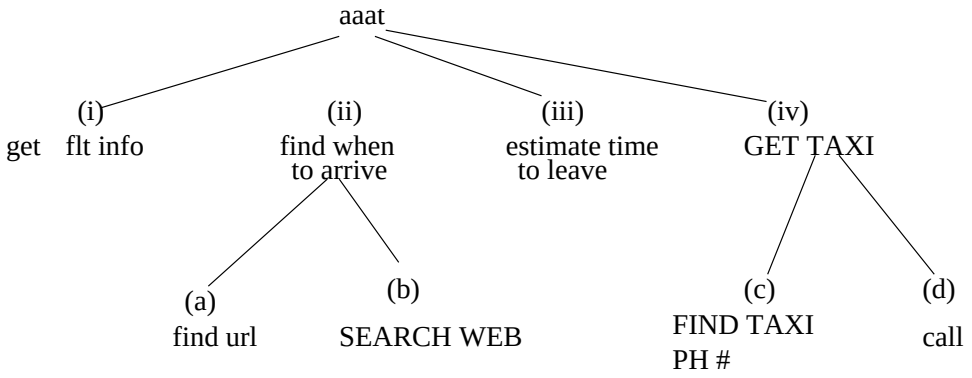


Figure 2: Recipe tree - the UPPER CASE actions are subcontracted to H.

of these complexities are taken up in a later section of the paper, including multiple recipes.

We assume that, as the various subactions in a recipe are performed, a vector of information is maintained and kept available for use during subsequent subacts. This information includes historical data on what has been done so far, as well as acquired information resulting from an action. For instance, in this example, the subaction "estimate time to leave" results in a time value that is placed into the vector and then used later when the taxi call is made. Since these details are at the agent level, they do not directly bear on the meta-theory. The overall tree of subactions is presented in Figure 2.

# 2 Syntax and Semantics

This section describes the syntax and the semantics of $L_M$, the metalanguage. In the first subsection we write the predicates of the meta-language and explain their use. The second subsection deals with the minimal model semantics that allows us to obtain, once the initial conditions and axioms are given, a unique minimal model of the theory showing the mental processes and activities of the agents as they change over time.

## 2.1 Sorted Language for Agent Intention

In this section we introduce the main predicates that we use in our work in the meta-language to characterize agent intentions. These will be constrained by the axioms in different ways. For each predicate we explain the use of the different attributes. We use a sorted language for comprehensibility and convenience so that, for example, agent names and times (for both of which we use positive integers) cannot be confused. Since the language is sorted, we use a convention for variables of different sorts, namely $t$,$i$,$j$,$k$ for time; $m$,$n$ for agent names, $a$,$b$,$c$ for actions, and $r$ for recipes. As needed, _ is used for the null element.

| Variables | Sort |
|---|---|
| t,i,j,k | time |
| m,n | agent names |
| a,b,c | actions |
| r | recipes |

Table 1: The convention for variables of different sorts.

We start with the two predicates for intention: $PotInt$ and $Int$, as well as AskedToDo ($ATD$) and Refrain ($Ref$). Basically, $PotInt$ represents a potential intention for an agent asked to perform an action. Under certain conditions a $PotInt$ will become an intention ($Int$).

The *context*, that is an argument of several of the predicates, keeps track of the tree structure of the recipes used. So, when an agent has a potential intention or an intention for an action, the context of the action, if there is one, is the parent node in the tree. For instance, in the example given in Figure 1, the context for $c2$ would be $b1$. For the root node, $a1$, the context is the null action _. So the context of a potential intention to do "find url" of the example of Figure 2 is "find when to arrive" and the context of a potential intention of "find when to arrive" is "aaat". Several predicates involve two agents, such as one agent assisting another agent; we allow these agents to be the same agent.

$ATD(t, n, m, b, a, t')$**:** At time $t$ agent $n$ asks agent $m$ to do action $b$ in the context of action $a$ at time $t'$. This predicate, which requires communication, is used both by the agent's owner who asks an agent initially to do a task, as well as by other agents as they request one another to do various tasks for them: this is subcontracting. There are two times involved, the time of the asking and the time that the action needs to be done. This will also be the case for several other predicates. In the example given above, writing $aaat$ for "arrange for arriving at airport on time", we could write $ATD(\text{14:30:00}, \_, G, aaat, \_, \text{15:00:00})$ to indicate that the owner (null agent) asked agent G (in the axioms we will assume for convenience that each agent is referred to by a number) at 2:30PM to start at 3PM the (root) action of arranging for arrival on time at the airport.

$PotInt(t, m, n, b, a, t')$**:** At time $t$ agent $m$ directly assisting agent $n$ (the null agent, in case there is no agent $n$) has the potential intention to do action $b$ in the context of action $a$ at time $t'$.

When $G$ is asked to arrange for arriving at the airport on time, it adopts a potential intention to do this task at the next time period. Assuming that a time period is one second, we will have $PotInt(\text{14:30:01}, G, \_, aaat, \_, \text{15:00:00})$ indicating that G will have a potential intention to start doing $aaat$ at 3PM. The task is done for the owner (hence the first _) and the arrange action is not a subaction in the context of another action (hence the second _).

$Int(t, m, n, b, a, t')$**:** At time $t$ agent $m$ directly assisting agent $n$ has the intention to do action $b$ in the context of action $a$ at time $t'$.

In our example, if $G$ or its assistants will adopt all the needed intentions to arrange for arriving at the airport on time in one minute, then the formula indicating this will be $Int(14{:}31{:}00, G, \_, aat, \_, 15{:}00{:}00)$.

$Ref(t, m, n, b, a, t')$**:** At time $t$ agent $m$ refrains agent $n$ from intending to do action $b$ in the context of action $a$ at time $t'$. As we show later, the effect of this request is that the agent in the next time period will no longer have the potential intention to do the action, in effect cancelling the task for the agent. We assume that $Ref$ includes the communication involved in an agent refraining another agent, but also allow for the case where $m = n$.

As noted in Section 1.2 we distinguish between *basic level* actions that are done by agents that can do them as primitive acts and $complex$ actions that consist of several subactions. The various subactions for achieving a complex action are specified in a recipe. The actions in a recipe may themselves be complex.

The following three predicates do not involve time in the sense of the previous predicates, because they refer to facts that are considered true for every time period.

$BL(a, d)$**:** $a$ is basic level, takes $d$ units of time to complete, and has no recipe. We will use $d = 1$ in the examples, but that is not necessary. In the example, we have $BL(call, 1)$.

$Rec(a, r)$**:** For action $a$, $r$ is the unique recipe (multiple recipes are discussed in Section 4).

$Mem(r, b, i, j, k)$**:** In recipe $r$, $b$ is the subaction which is the $i$'th member of the recipe starting at relative time $j$ and ending at relative time $k$ (i.e. relative, or as an offset, to the time at the beginning of the action). For example, the formula $Mem(r_1, get\_flt\_info, 1, 1, 2)$ states that $get\_flt\_info$ is the first action for recipe $r_1$ and that it starts at time 1 and ends at time 2, where these times are relative to the initiation of the recipe.

In the next group we deal with other relevant concepts about agents: their beliefs, abilities to do actions, and means of communication with other agents.

$Bel(t, n, f)$**:** At time $t$ agent $n$ believes the statement expressed by formula $f$ (i.e., $f$ is the name of the statement).

$CanDo(t, n, a)$**:** At time $t$ agent $n$ can do (or at least start) action $a$.

For example, if at 14:31, $G$ believes that it can do at 15:00 the action $aaat$, then the formula indicating this will be $Bel(14{:}31{:}00, G, "CanDo(15{:}00{:}00, G, aaat)")$.

$Tell(t, n, m, f)$**:** At time $t$ agent $n$ tells agent $m$ the statement expressed by the formula $f$. The formulas are the formulas of the agent language, $L_A$. These are the formulas that are meaningful to the agents. Each such formula has a name (its quoted version) in the meta-language $L_M$.

For example, $Tell(14{:}35{:}00, H, G, "Int(14{:}35{:}00, H, G, search\_web, \_, 15{:}06{:}00)")$ means that at 14:35, agent $H$ tells agent $G$ that it intends to search the web at 15:06.

Two predicates deal with the agents actually doing the actions, both the initialization and completion.

$Ini(t, m, n, a)$**:** At time $t$ agent $m$ directly assisting agent $n$ initiates action $a$.

For example, $Ini(15{:}00{:}00, G, \_, aaat)$ indicates that at time 15:00 agent $G$ initiates the action $aaat$.

$Done(t, a)$**:** At time $t$ action $a$ has just been done successfully.

There is a predicate indicating that a subaction failed to be done stopping the performance of the action.

$Stop(t, m, n, b, a)$**:** At time $t$ agent $m$ directly assisting agent $n$ is instructed to stop action $b$ in the context of action $a$. As with $Ref$, $Stop$ includes the appropriate communication.

Both $Stop$ and $Ref$ are applied to terminate an action. However, $Ref$ is applied to terminate an action during the intention formation process, while $Stop$ is applied during the execution of the action.

Finally we include two predicates used in an integrity constraint.

$Prefer(t, n, a, b)$**:** At time $t$ agent $n$ prefers to do action $a$ over action $b$.

$Conf(t, n, a, b)$**:** At time $t$ for agent $n$ action $a$ conflicts with action $b$.

Communication is important in team activity in general, and in subcontracting in particular. We apply one explicit communication predicate– $Tell$. In addition, as discussed above, several other predicates require communication: $ATD$, $Ref$ and $Stop$. The axioms in which these predicates appear could have been stated using the $Tell$ predicate, but introducing special predicates to these communication related activities simplified our meta-logic. Communication may also be necessary in the initialization process. We assume that an agent can observe whether an action, possibly performed by another agent, preceding its own action has been done, and comment on the need for further communication when such observation is not possible.

## 2.2 Minimal Model Semantics

The axioms of the meta-theory will typically have the form $\forall\vec{x}(\alpha_1 \& \ldots \& \alpha_n \rightarrow \beta)$, where $\vec{x}$ is the sequence of variables in the formula, each $\alpha_i$ and $\beta$ is an atom, and the time of the relevant $\alpha_i$s is $t$ and the time of $\beta$ is $t + 1$. The axioms of the meta-theory can be shown to be consistent by constructing a model for them. In [18] Page 361 we prove that the meta-theory given in that paper is consistent by showing how to construct a minimal model for it. The proof does not depend on the specific axioms given there, only on the form of the axioms. Hence the same process can be used to construct a model for the meta-theory presented in this paper. In general, there will be many models for such a first-order theory that have only a tenuous relationship to the application of our interest. We wish to interpret the axioms in such a way that for any instantiation of the formula, where all the $\alpha_i$ are true in the model, $\beta$ is derived.

This way, we will be able to use the axioms to derive in the meta-theory the beliefs, intentions, and actions of the agents at time $t+1$, given information about the situation at time $t$.

In addition to axioms, we allow integrity constraints in the meta-theory as well. Integrity constraints have the form $\forall\vec{x}\neg(\alpha_1\&\ldots\&\alpha_n)$. Integrity constraints are not used to derive results about agent beliefs, intentions, or actions over time; their purpose is to eliminate possible models that do not make sense. Typically, integrity constraints involve time only in a general way. For example, one integrity constraint states that for every action the ending time must come after the starting time.

Consider now that first-order logic formulas may be written in many logically equivalent forms. For example, a formula of the form
$\forall\vec{x}(\alpha_1\&\ldots\&\alpha_n\rightarrow\beta)$
can be written in a logically equivalent form as
$\forall\vec{x}(\alpha_1\&\ldots\&\alpha_{n-1}\&\neg\beta\rightarrow\neg\alpha_n)$.
For our purpose, however, it would not be useful to use this version of the axiom deriving something about the agent at time $t$ from something that did not happen at time $t+1$. However, these two formulas have the same models.

This analysis prompts us to restrict consideration to only some of the models of the meta-theory. The first restriction is to consider only Herbrand models, i.e., models that contain only elements that are ground terms of $L_M$. The second restriction is to consider only minimal models where we minimize each predicate for one time period at a time starting with time $0$ and then proceeding by induction on time. Because of the structure of the axioms, there will be only one such model. We can construct this model by a process that is similar to the construction of the unique perfect model of a locally stratified theory in logic programming. See [18] for details. Such a theory is also called XY-stratified and discussed on page 247 of [58].

In order to have a sharp sense of how our agents behave, we simply define them to be processes that behave as in this unique minimal model. This has the desired effect of providing certain converses of the axioms as true statements in this model. In effect we are making an assumption of complete information (akin to a closed-world assumption) about agent behavior; for example an agent comes to have a particular potential intention at time t, if and only if our axioms require that agent to have that potential intention at that time, and thus this potential intention will be true in the unique minimal model. Similarly, for example, agent $G$ will ask agent $H$ at time 11 to get a taxi, iff $ATD(11, G, H, get\_taxi, aaat, 58)$ is true in the minimal model. This is a restriction of sorts; after all, much of commonsense reasoning involves situations that are usually taken to be too complex to fully axiomatize. But our view is to suppose that the complexities primarily arise from the external world and that the agent behaviors are completely characterized by the axioms, once the external inputs are given.

Later, by studying the minimal model under certain initial conditions, known facts about agent beliefs and abilities and the structure of recipes, we will prove several meta-theorems. These theorems typically state that given the beliefs and abilities of agents, and given "enough" time as defined by a formula involving information about the recipes, the agent or agents will be able to plan and execute the requested action.

# 3 Modeling Multiple Agents with Single-Recipe Actions

In this section we model agents that have only one way, a single recipe, to perform actions. We present a theory $\mathcal{T}$ over $L_M$ which consists of the axiom schemas that describe the desired intentions and behavior of a team of cooperative agents and how their intentions change over time. We do not present the basic axioms of belief, time etc. that are given in [18]. We also state and prove several general theorems about the meta-theory. These theorems show that, under the appropriate conditions, the meta-theory characterizes the agent activities in a reasonable manner.

## 3.1 Axioms of the Meta-theory

We divide the axioms into three groups: the intention formation axioms, the subcontracting axioms, and the performing axioms. We also have integrity constraints. Before specifying the axioms, we sketch the general scenario for the agent potential intentions, intentions, subcontracting and doing actions. We note that in any application of the theory there will be additional domain specific facts (axioms) that need to be invoked.

When an agent is asked to do an action $a$ at a particular time, in the next time period it adopts a potential-intention to do it. If the agent believes that it can do the action $a$, then in the basic level case, in the next time period it will adopt an intention to do it. In the case of a complex action, it will look for a recipe and in the next time period will adopt potential intentions for all the immediate subactions, some of which may themselves be complex, of the recipe.

If after adopting a potential intention the agent comes to believe that it cannot do the action $a$, in the next time period it will ask another agent, that it believes can do $a$, to do $a$. If at a particular time period an agent has a potential intention to do a complex action, it will adopt in the next time period an intention to do the action if for each subaction in the recipe it either adopted an intention to do it or it has found another agent with an intention to do it. Even after the initial adoption of an intention, during the waiting phase, the process of checking the needed associated intentions and beliefs and the adopting of an intention will be repeated while the agent still has the potential intention and the time to do the action has not passed.

Each request to perform an action has a time associated with it. If all the needed intentions have been adopted for an action in time, then at the specified time the agent will initiate the action. This will lead to the performance of all the basic actions in the recipe tree at the appropriate times, assuming that the agents can actually do the actions. Failure may occur either in the intention formation phase, the waiting phase or during the execution phase. Each such failure should be reported to the relevant agents, i.e., the assisting agents and the asking agent.

In writing the axioms we use the convention that all free variables are assumed to be universal quantified. The axioms typically have the form $\alpha \rightarrow \beta$ where the time of $\alpha$ is $t$ and the time of $\beta$ is $t + 1$. This is because the axioms characterize the way the mental state of the agent and the state of the world change over time. This way we capture the property that agent reasoning and actions take time.

Additional convention has been introduced to simplify the axioms. When a given proposition is included in the databases of all the agents, the belief predicates will be

omitted, e.g., we write $BL(a, d)$ instead of $\forall m \forall t Bel(m, t, "BL(a, d)")$. The belief predicate is explicitly specified only when the meta theory refer to the belief of a specific agent, e.g., $Bel(t, m, "CanDo(t', m, b)")$.

### 3.1.1 Intention Formation Axioms

We list here the axioms involved in agent intention formation. Although the subcontracting axioms are not included here, we write the axioms in a general way so that they are applicable both to the case where a single agent does all the work and the case of multiple agents. We explain in English the use of each axiom before writing it in logic. At the end of the subsection we write some applications of the axioms using our example.

**A1. Asked To Do Becomes Potential Intention**

If agent $n$ asks agent $m$ to do an action, agent $m$ adopts a potential intention for it. When the first agent is initially asked to do the action, $n = \_$ and $m$ is the agent.

$$ATD(t, n, m, b, a, t') \ \& \ t + 1 < t' \rightarrow PotInt(t + 1, m, n, b, a, t')$$

**A2. Inheritance of Potential Intention**

We distinguish between the inheritance of the potential intention to do an action $b$ and the adoption (and later inheritance) of the potential intentions of the subactions in the recipe for $b$.

**A2.1: self** If an agent has a potential intention for $b$ and is not refrained from doing $b$, then it will inherit the potential intention.

$$PotInt(t, m, n, b, a, t') \ \& \ \neg \exists m' Ref(t, m', m, b, a, t') \ \& \ t + 1 < t' \rightarrow PotInt(t + 1, m, n, b, a, t')$$

**A2.2: child** If an agent has a potential intention for $b$ and believes that it can do $b$, it will get a potential intention for every subaction of $b$ for the appropriate time (based on the times of the subactions in the recipe).

$$PotInt(t, m, n, b, a, t') \ \& \ Bel(t, m, "CanDo(t', m, b)") \ \& \ Rec(b, r) \ \& \ Mem(r, c, i, j, k) \ \& \ t + 1 < t' \rightarrow PotInt(t + 1, m, m, c, b, t' + j)$$

**A3. Potential Intention Becomes Intention**

If an agent has a potential intention for $b$ (in the context of $a$) and believes that it can do $b$ and has the intention to do all the subactions (if any) in the recipe for $b$, then it will get the intention to do $b$. In the next subsection we will present a more general version of this axiom including subcontracting as B3.

$$PotInt(t, m, n, b, a, t') \ \& \ Bel(t, m, "CanDo(t', m, b)") \ \& \ t + 1 < t' \ \& \ \forall r, c, i, j, k (Rec(b, r) \ \& \ Mem(r, c, i, j, k) \rightarrow Int(t, m, m, c, b, t' + j)) \rightarrow Int(t + 1, m, n, b, a, t')$$

**A4. Inheritance of Refrain**

Refrains cancel potential intentions. Thus, as in the potential intention case, the axiom associated with the inheritance of refrains of an action $b$ has two parts: one has to do with the refrain of $b$ itself, and one with respect to the subactions in the recipe for $b$.

**A4.1: self** Refrain is inherited by an action unconditionally.

$$Ref(t, m, n, b, a, t') \ \& \ t + 1 < t' \rightarrow Ref(t + 1, m, n, b, a, t')$$

**A4.2: child** Refrain of an action is inherited by all the subactions in the recipe.

$$Ref(t, m, n, b, a, t') \ \& \ Rec(b, r) \ \& \ Mem(r, c, i, j, k) \ \& \ t + 1 < t' \rightarrow Ref(t + 1, n, n, c, b, t' + j)$$

In the examples, for convenience, we write the time values as integers instead of as clock times but leave the agent names as G and H. Returning to the main example, suppose that at time 10 agent G is asked to do $aaat$ at time 50. In the meta-language this is represented as the statement $ATD(10, \_, G, aaat, 50)$.

- Axiom A1 entails $PotInt(11, G, \_, aaat, \_, 50)$.
- A2.1 entails $PotInt(12, G, \_, aaat, \_, 50)$ assuming that the corresponding $Ref$ is not invoked.

Now, assume $Bel(11, G, "CanDo(50, G, aaat)")$,
$Rec(aaat, r_0)$ and $Mem(r_0, estimate\_time\_to\_leave, 3, 6, 7)$ are already known, where $r_0$ is the recipe of Figure 2 and $estimate\_time\_to\_leave$ is the third subaction that is supposed to start at 6 time units after the beginning of $aaat$ and ending at 7 units after it.

- A2.2 entails $PotInt(12, G, G, estimate\_time\_to\_leave, aaat, 56)$.
- A3 entails $Int(13, G, G, estimate\_time\_to\_leave, aaat, 56)$, assuming $Bel(12, G, "CanDo(56, G, estimate\_time\_to\_leave)")$ and knowing $BL(estimate\_time\_to\_leave, 1)$.

### 3.1.2 Subcontracting Axioms

Subcontracting is the process that leads to an agent adopting an intention to do an action motivated by a request from another agent. It is initiated by a request of the contracting agent to the assisting agent, continues with the adoption of the potential intentions and later an intention by the assisting agent. The subcontracting process can terminate by a failure either during the intention formation process or waiting period (e.g., refrain). Both the adoption of the relevant intention by the assisting agent and the failures are reported to the contracting agent.

We write four axioms for subcontracting between agents. In order to keep the numbering analogous to the numbering of the $A$ axioms, we start the numbering with $B2$. Two of the axioms involve explicit communication between agents.

**B2. Subcontracting an Action to an Agent**

This axiom deals with subcontracting an action to one of the agents that can do it. This happens when an agent has a potential intention to do an action, but does not believe that it can do it. Unless it will ask another agent, this potential intention will not become an intention. The antecedents of this axiom are that the agent has a potential intention to do an action and does not believe that it can do it and believes that agent $m$ is the "first" agent that can do it and it has not previously asked $m$ to do it. The consequence is that the agent will ask $m$ to do the action.

$$\begin{aligned}
&PotInt(t, n, n', b, a, t') \ \& \ \neg Bel(t, n, "CanDo(t', n, b)") \ \& \\
&Bel(t, n, "CanDo(t', m, b)") \ \& \\
&\forall m'(Bel(t, n, "CanDo(t', m', b)") \ \rightarrow \ m \leq m') \ \& \\
&\forall t"(t" < t \ \rightarrow \ \neg ATD(t", n, m, b, a, t')) \ \& \ t + 1 < t' \\
&\qquad\qquad \rightarrow \ ATD(t + 1, n, m, b, a, t')
\end{aligned}$$

**B3. Potential Intention Becomes Intention (Subcontracting Version)**

If an agent, assisting another agent, has a potential intention to do an action and believes that it can do the action (possibly with help) and if each subaction in the recipe is either intended by the agent or an assisting agent, then the agent will have the intention to do the action and tell that intention to the agent it is assisting.

$$\begin{aligned}
&PotInt(t, n, n', b, a, t') \ \& \ Bel(t, n, "CanDo(t', n, b)") \ \& \\
&Rec(b, r) \ \& \ t + 1 < t' \ \& \\
&\forall c, i, j, k(Mem(r, c, i, j, k) \rightarrow (Int(t, n, n, c, b, t' + j) \vee \\
&\qquad\qquad \exists m Tell(t, m, n, "Int(t, m, n, c, b, t' + j)"))) \\
&\qquad \rightarrow \ Int(t + 1, n, n', b, a, t') \ \& \ Tell(t + 1, n, n', "Int(t + 1, n, n', b, a, t')")
\end{aligned}$$

**B4. Inheritance of Refrain By an Assisting Agent**

If an agent is refrained from doing an action and this agent has an assisting agent, then the assisting agent will be refrained from doing the action.

$$\begin{aligned}
&Ref(t, n', n, b, a, t') \ \& \ Rec(b, r) \ \& \ Mem(r, c, i, j, k) \ \& \\
&\exists t"(t" < t \ \& \ ATD(t", n, m, c, b, t' + j)) \ \& \ t + 1 < t' + j \\
&\qquad\qquad \rightarrow \ Ref(t + 1, n, m, c, b, t' + j)
\end{aligned}$$

**B5. Communication of Assisting Agent About Refrain**

If an agent has a potential intention to do an action for a requesting agent and is refrained from doing it, then the agent will tell the requesting agent that it does not have the intention to do the action and the requesting agent will believe that the asssisting agent cannot do the action.

$$\begin{aligned}
&PotInt(t, m, n, b, a, t') \& Ref(t, m', m, b, a, t') \& \\
&\forall t"(t" < t \rightarrow \neg Tell(t", m, n, "\neg Int(t", m, n, b, a, t')")) \\
&\rightarrow Tell(t{+}1, m, n, "\neg Int(t{+}1, m, n, b, a, t')") \& Bel(t{+}2, n, "\neg CanDo(t', m, b)")
\end{aligned}$$

In both axioms B4 and B5 there is communications. In B5 the communication is stated explicitly while in B4 it is associated with the refrain. In addition, when an agent is refrained from doing an action there is a difference between how it influences the assisting agents of subactions and the requesting agent, as is reflected in axioms B4 and B5 respectively. All the assisting agents are also refrained since there is no need any more for them to carry out the subactions. The requesting agent is told about the refrain and it will ask another agent, if possible, or refrain.

Continuing with our example, suppose that at time 12 G has a potential intention to do $get\_taxi$ and believes that H can do this subaction. In our meta-language this is represented as $PotInt(12, G, \_, get\_taxi, aaat, 58)$ and $Bel(12, G, "CanDo(58, H, get\_taxi)")$. Also suppose that G does not believe that it can do this subaction and has not asked H previously to do it. Thus the theory contains $\neg Bel(12, G, "CanDo(58, G, get\_taxi)")$ and $\neg ATD(t, G, H, get\_taxi, aaat, 58)$ for every $t < 12$. Then,

- B2 entails $ATD(13, G, H, get\_taxi, aaat, 58)$.

### 3.1.3 Performing Axioms

Next we present four axioms involving agents doing actions. The first shows how the root of the recipe tree is initiated. When a basic level action is initiated by an agent, it gets done in $d$ time units if the agent can do it. For a complex action, the agent must start by initiating the root of the recipe tree. Each node will have to be initiated in turn. Several agents may be involved in doing various subactions. We use the initiation of complex actions as a bookkeeping device to make sure that all the subactions get done at the proper time and in the right order.

**C1. Initiation of Requested Action**

If an agent has an intention to do an action for a second agent at the next time period, which is not a subaction of an action the agent intends to do, then the agent will initiate the action. A special case arises when the second agent is _; this initiates the root action.

$$Int(t, m, n, b, \_, t + 1) \rightarrow Ini(t + 1, m, n, b)$$

**C2. Inheritance of Initiate for First Subaction**

If an agent initiates an action, it will initiate the first subaction in the recipe in the next time step.

$$Ini(t, m, n, a) \ \& \ Rec(a, r) \ \& \ Mem(r, b, 1, 1, k) \rightarrow Ini(t + 1, m, m, b)$$

**C3. Inheritance of Initiate for Later Subaction**

If the previous subaction is done at the right time and the agent doing action $a$ has subcontracted this subaction to another agent, then this second agent will initiate the next subaction. The special case is where the same agent does this later subaction. Note that we assume an agent can observe the performance of

the actions that are needed for the initiation of its own actions, and that its observations lead to beliefs. If not, further communication is needed, as discussed below.

$$Ini(t, m, n, a) \ \& \ Rec(a, r) \ \& \ Mem(r, b, i+1, j', k') \ \&$$
$$Mem(r, c, i, j, k) \ \& \ Bel(t + j' - 1, m', "Done(t + k, c)") \ \&$$
$$Int(t + j' - 1, m', m, b, a, t + j')$$
$$\rightarrow \ Ini(t + j', m', m, b)$$

**C4. Done for Actions**

**C4.1: basic level** If an agent initiates a basic level action at time $t$ and can do it then the action will get done.

$$BL(a, d) \ \& \ Ini(t, m, n, a) \ \& \ CanDo(t, m, a)$$
$$\rightarrow \ Done(t + d, a)$$

**C4.2: complex** If an agent initiates a complex action and all its subactions in the recipe get done, then the action will get done. (Note: $b$ is the last subaction of $a$.)

$$Ini(t, m, n, a) \ \& \ Rec(a, r) \ \& \ Mem(r, b, i, j, k) \ \&$$
$$\forall r, c, i', j', k'(Mem(r, c, i', j', k') \rightarrow (i' \leq i \ \& \ Done(t + k', c)))$$
$$\rightarrow Done(t + k, a)$$

**C5. Action Performance Observed:** If an action is done, then all agents come to believe this fact (via observations).

$$Done(t, a) \qquad \rightarrow \ Bel(t, n, "Done(t, a)")$$

Although for simplicity we assume that all agents observe all actions; in fact, only two agents, the requesting agent and the agent intending to do the next action really need to know. In multiagent systems in which agents can't observe the activities of other agents, communication is needed to establish the performance of actions. There are many communication models that can be applied, such as broadcasting, blackboards and directed messages. For each communication model C5 could be replaced by appropriate axioms.

Again, continuing with our example from the previous subsections, Suppose at time 49 G has the intention to do $aaat$, represented as $Int(49, G, \_, aaat, \_, 50)$. Then,

- C1 will entail $Ini(50, G, \_, aaat)$, thereby initiating the root action.
- C2 will then entail $Ini(51, G, G, get\_fl\_info)$.

Considering another subaction, suppose that estimate_time_to_leave is a basic level action taking one unit of time that G initiates and can do at time 58, written as, $BL(estimate\_time\_to\_leave, 1)$,
$Ini(58, G, G, estimate\_time\_to\_leave)$, and $CanDo(58, G, estimate\_time\_to\_leave)$, then

- C4.1 entails $Done(59, estimate\_time\_to\_leave)$ and
  C5 entails $Bel(59, H, "Done(59, estimate\_time\_to\_leave)")$

- From $Ini(50, G, \_, aaat)$, $Rec(aaat, r_0)$, $Mem(r_0, get\_taxi, 4, 10, 14)$,
  $Mem(r_0, est\_time\_to\_leave, 3, 8, 9)$, $Bel(59, H, "Done(59, estimate\_time\_to\_leave)")$,
  and $Int(59, H, G, get\_taxi, aaat, 60)$, C3 entails $Ini(60, H, G, get\_taxi)$.

That is, H will initiate the get_taxi subaction at time 60 for G.

#### 3.1.4 Axioms for Failure to Perform

Here we present three axioms for the case when the performance of an action fails. Failure begins when an agent fails to perform a basic level action, as indicated by the negation of the appropriate Done. This leads to a Stop action process that will release the agents waiting to perform various actions from their commitments, that is, their intentions. This is accomplished in the following way. In the recipe tree the Stop action process is propagated upwards. At the same time the Stop action becomes a Refrain going down in the recipe tree along the branches that come after the branch that failed.

**D1. Initiate Stop Action in case of Failure**
If an agent fails to do a basic level action, it tells the requesting agent and initiates a stop action process on the node of the recipe tree above this basic level action.

$$BL(b, d)\ \&\ Ini(t, m, n, b)\ \&\ \neg Done(t + d, b)\ \&\ Int(t - 1, m, n, b, a, t)\ \&$$
$$\exists t' n' a' (t' < t - 1\ \&\ Int(t', n, n', a, a', t' + 1))$$
$$\rightarrow Stop(t + d + 1, n, n', a, a')$$

**D2. Stop Action Propagated Up**
If an agent initiates a stop action on a node, the stop action is propagated to the parent node.

$$Stop(t, m, n, b, a)\ \&\ \exists t' (t' < t\ \&\ Int(t', n, n', a, a', t' + 1))$$
$$\rightarrow Stop(t + 1, n, n', a, a')$$

**D3. Stop Triggers Refrain Down**
A stop action triggers a refrain for those subactions that come after the branch that failed.

$$Stop(t, m, n, b, a)\ \&\ Rec(b, r)\ \&\ Mem(r, c, i, j, k)\ \&$$
$$t' > t\ \&\ Int(t, m', m, c, b, t')$$
$$\rightarrow Ref(t + 1, m, m', c, b, t')$$

The above actions apply both the $Stop$ predicate and $Ref$. $Stop$ is used for actions that have been initiated, while $Ref$ is used for actions that one of the agents intends to perform, but their performance time has not arrived yet.

### 3.1.5 Integrity Constraints

In addition to the axioms we can include integrity constraints in our theory. Integrity constraints differ from axioms in that they eliminate certain models from consideration, models that would not make sense given the meaning of our predicates. We give only a few integrity constraint here, but in other contexts more may be needed. We write the integrity constraints using logic programming notation.

**IC1. Preference among Conflicting Actions**

An agent that prefers action $a$ to conflicting action $b$ cannot both potentially intend to do $a$ and believe that it can do $b$ at the same time. (Agents are assumed to have preferences among conflicting actions.)

$$\leftarrow Conf(t,n,a,b)\ \&\ Prefer(t,n,a,b)\ \&\ PotInt(t,n,m,a,c,t')\ \&\ Bel(t,n,"CanDo(t',n,b)")$$

**IC2. Consistency of Recipe Subactions**

In a recipe the starting time of the i+1st subaction must be greater than the ending time of the ith subaction.

$$\leftarrow\ Rec(a,r)\ \&\ Mem(r,b,i,j,k)\ \&\ Mem(r,b',i+1,j',k')\ \&\ j' \leq k$$

**IC3. Consistency of Recipe Timing**

For any member of a recipe the starting time must be less than the ending time.

$$\leftarrow Mem(r,b,i,j,k)\ \&\ k \leq j$$

**IC4. Uniqueness of Recipe Information**

There cannot be more than one $i^{th}$ subaction in a recipe.

(1) $$\leftarrow Rec(a,r)\ \&\ Mem(r,b,i,j,k)\ \&\ Mem(r,b',i,j,k)\ \&\ b \neq b'$$

A subaction may appear only once in a recipe. This constraint is not included in the case of repetitive actions in a recipe.

(2) $$\leftarrow Rec(a,r)\ \&\ Mem(r,b,i,j,k)\ \&\ Mem(r,b,i',j,k)\ \&\ i \neq i'$$

A subaction must start at a unique time.

(3) $$\leftarrow Rec(a,r)\ \&\ Mem(r,b,i,j,k)\ \&\ Mem(r,b,i,j',k)\ \&\ j \neq j'$$

A subaction must end at a unique time.

(4) $$\leftarrow Rec(a,r)\ \&\ Mem(r,b,i,j,k)\ \&\ Mem(r,b,i,j,k')\ \&\ k \neq k'$$

### 3.1.6 Parallel Actions

Up to now we have assumed that the actions in a recipe must be done in a serial order. So, for example, in our running example of $aaat$ (arrange for arriving at airport on time) the four subactions must be done in the order given in Figure 2. In particular,

"find when to arrive" must be done before "estimate time to leave". However, there are cases where several actions can be done in parallel. Continuing with the travel example, we may have an action "pack suitcase" which consist of "get suitcase", "gather objects to be taken", and "put objects in suitcase". The "put objects in suitcase" subaction must be done last, but the first two subactions can be done in either order or in parallel. We now show that our framework can handle recipes with parallel actions; only a few small changes have to be made to the axioms.

The first change is in the uniqueness of recipe information constraints. **IC4** (1) must be changed by changing $i$ to 1. That is, we allow several different subactions, say $b_{i1}, b_{i2}, \ldots, b_{ik}$ to be done in parallel after the subaction $b_{i-1}$. So we consider the $i^{th}$ subaction to be a set of subactions. We do not allow a subaction to appear more than once in such a set. However, the first subaction still must be treated in a special way because the agent that believes it can do it believes that it can do the action itself (possibly with help). Hence there can be only one first subaction. We may also remove **IC4** (3) and (4) in case different parallel actions may start and end at different times.

The second change is the modification of axiom **C3**. In the original version of this axiom the $i + 1^{st}$ subaction is initiated after the $i^{th}$ subaction is done and the assumption is that there is a single $i^{th}$ and $i + 1^{st}$ subaction. In the parallel version we need to write, see below, that the $i + 1^{st}$ subactions are initiated after all the $i^th$ subactions are done.

$$Ini(t, m, n, a) \ \& \ Rec(a, r) \ \& \ Mem(r, b, i + 1, j', k') \ \&$$
$$\forall c, i, j, k(Mem(r, c, i, j, k) \ \rightarrow \ Bel(t + j' - 1, m', "Done(t + k, c)") \ \&$$
$$Int(t + j' - 1, m', m, b, a, t + j')$$
$$\rightarrow \ Ini(t + j', m', m, b)$$

## 3.2 Consistency and Meta-theorems

Consider a meta-language $\mathcal{L}$ and a theory $\mathcal{T}$ in $\mathcal{L}$ that contains the axioms we presented in Section 3.1 as well as additional specific axioms related to the agents, such as the beliefs of the agents.

We first prove that any such theory $\mathcal{T}$ is consistent.

**Theorem 1** *$\mathcal{T}$ has a model and is therefore consistent.*

**Proof:** The proof is essentially the same as that of Theorem 1 of [18], so we omit some of the details. We note that most of the predicates have as their first argument a time value $t$. Those predicates that do not contain such a time value, such as $BL$, represent statements that are always true. Suppose that we are given a specific language $\mathcal{L}$. The atoms of the Herbrand universe for $\mathcal{L}$ can be divided into groups based on the time value $t$; use $t = 0$ for atoms without such a time value.

The construction of the model uses the statements of $\mathcal{T}$. These statements are either atoms or implications where the time component of the consequent is greater than or equal to the time components in the antecedent. Start with the statements whose time value (either explicitly or implicitly) is 0. Proceed by induction on these time values, thus assuming that at time $t+1$ all the atoms with time $t$ have already been constructed. Then, use the axioms to generate the atoms at time $t + 1$ in those cases where by the

appropriate substitutions the antecedent of the axiom is already true in the model. This construction yields a model that is also a minimal model and which we take to be the intended model. Every positive fact in this model must be there because of the applications of the axioms. ▌

We take this minimal model as the intended model. By studying this minimal model under certain initial conditions and known facts about agent beliefs and abilities and the structure of recipes, we can prove various results. The proofs consist of tracing the steps of the agents over time in the model. We include three results here: two positive, one negative. The two theorems show that if there is "enough" time allowed for the intention formation process associated with a complex action all of whose subactions a single agent can do (Theorem 2) or subcontract some to other agents (Theorem 3), then the agent will do it. The Proposition shows that if one basic level action in the recipe of a complex action cannot be done by any agent, then no agent will get the intention to do the complex action. There are some models of the meta-theory where this result fails, but it is true in the intended model.

**Theorem 2** *Suppose an agent is asked at time $t$ to do an action $a$ starting at time $t + s$, and suppose that the recipe tree for $a$ has height $h$, takes $k$ units of time, and $s \geq 2h + 3$. Further suppose the agent believes that it can do each subaction (at the needed time) itself, is not refrained at any time from doing the action or any subaction, and that it in fact can do all of them. Then the agent will complete the task at time $t + s + k$. Moreover, any value for $s$ smaller than $2h + 3$ will not be sufficient to start the action at time $t + s$.*

**Proof :**

It suffices to show that intention formation takes at most $2h+2$ units of time because then the agent can start doing the action at time $t + s \geq t + 2h + 3$ and since it can do each step, it will finish at time $t + s + k$. We do this by showing abbreviated versions of the key formulas that become true as time changes: we omit the agent name (there is only one agent) and the times. We also omit the inherited formulas but add some comments about them at the end. We also do not show other true formulas, such as $BL(z1)$, that are assumed to hold for all times.

**t** $ATD(a)$

**t+1** $PotInt(a)$

**t+2** $PotInt(b1)$, $PotInt(b2)$, ... the children node of $a$

**...**

**t+h+1** $PotInt(z1)$, $PotInt(z2)$, ... the nodes at the bottom level

**t+h+2** $Int(z1)$, $Int(z2)$, ...

**...**

**t+2h+1** $Int(b1)$, $Int(b2)$, ...

**t+2h+2** $Int(a)$

**...**

**t+s** $Ini(a)$

**t+s+1** $Ini(b1)$

**...** The times for the Ini and Done of the nodes depend on the tree structure

**t+s+k** $Done(a)$

We note that $PotInt(a)$ will actually hold starting at time $t+1$ and ending at time $t+s-1$. In fact, in this case all the potential intentions and intentions will hold from their earliest time as indicated above (e.g. $t+2$ for $PotInt(b1)$, $t+h+2$ for $Int(z1)$) until $t+s'-1$, where the subaction is supposed to be done at time $t+s'$. Since $s \geq 2h+3$, the agent will be able to initiate the action at time $t+s$. Because $Ini(a)$ must occur after $Int(a)$, for any $s$ less than $2h+2+1 = 2h+3$, there will not be enough time for the agent to obtain $Int(a)$ and hence cannot initiate the action at time $t+s$. ∎

In particular, in the $aaat$ example $h = 2$, hence if only one agent is doing all the subactions, 6 time units must be allocated for the intention formation process, so that the work can start at 7 units after the initial request for the action.

The next result generalizes the previous one to the case of multiple agents with subcontracting. We note, however, our assumption that each subcontracting between agents is successful. This may be a reasonable assumption if each agent has knowledge of other agents' abilities (and their beliefs about them) and the agents are not busy doing other tasks. We will show later what happens when subcontracting is not always successful.

**Theorem 3** *Suppose an agent is asked at time $t$ to do an action $a$ starting at time $t+s$, suppose that the recipe tree for $a$ has height $h$ and takes $k$ units of time, and $s \geq 4h+5$. Further, suppose for each subaction the agent either believes and can do it or can successfully subcontract it to the first agent that it believes can do it (at the needed time), and none of the agents is refrained at any time from doing an action or any subaction, and in fact the intending agent can always do the action or subaction. Then the agent, with the assistance of the subcontracting agents, will complete the task at time $t+s+k$. Moreover, any value of $s$ smaller than $4h+5$ will not, in general, be sufficient to start the action at time $t+s$.*

**Proof :**

We use the proof of Theorem 2 and show where this proof differs from it. What may happen at any node is that the agent subcontracts the subaction to another agent. Using Axiom B2 takes one step and then the second agent gets a potential intention by using A1. Thus at each of the $h+1$ levels at most two time steps must be added, so in the worst case the potential intentions at the bottom level are reached at time $t+3h+3$ instead of $t+h+1$. Going back up the tree takes the same amount of time as before,

that is, $h + 1$ time units. Hence the intention for $a$ will be reached at time $t + 4h + 4$. The bound for $s$ is needed in case subcontracting takes place at each level.

∎

In our running example of $aaat$ with 2 agents and $s = 2$, $4h + 4 = 12$ units of time must be allocated for the intention formation process, in order to initiate the action at time $t + 13$.

**Proposition 1** *If there is a basic level action $b$ in the recipe for $a$ that no agent believes it can do, then no agent will get an intention to do $a$.*

**Proof :**
No agent will ever get an intention to do $b$ and hence to do $a$.

∎

# 4 Modeling Multiple-Recipe Actions

The previous section dealt with the intention formation and subcontracting (and performing) of actions, where each action has a single recipe. This leads to a single recipe tree for a complex action. In this section we generalize the work for intention formation and subcontracting to the case where actions (may) have multiple recipes. In this case we may think of the recipe tree as an and-or tree with alternating levels of recipes and actions where the recipes for an action are joined by "or" and the actions of a recipe are joined by "and".

The key difference in reasoning between the single recipe case and the multiple recipe case is that the recipe tree becomes a complex and-or tree in the second case. When an agent finds that one recipe will not work, it must find and apply another one. When all the recipes for an action don't work, the agent must backtrack in the tree to the parent action of that action and find another recipe for it. Another issue is that if a subcontracting agent is determined as unable to do a task, another agent must be subcontracted. A second backtracking must then be introduced to represent the backtracking between the subcontracting agents in addition to the backtracking between recipes.

In order to enhance the presentation we deal separately with the case of a single agent and multiple agents. We show the details for the single agent case, in particular, the handling of recipe backtracking, and sketch the modifications, including the backtracking among agents, required by multiple agents.

## 4.1 The Single Agent Case

As we deal with the case of a single agent, the agent names are not needed. We omit them everywhere in order to make the axioms easier to understand. However, some predicates now need to include the recipe name. For convenience we start by listing the predicates used in this section, before presenting and explaining the axioms.

### 4.1.1 The Predicates

The following predicates are important for single agent intention formation with multiple recipes.

$ATD(t, a, t')$**:** At time $t$ the agent is asked to do $a$ at time $t'$.

$PotInt(t, b, a, r, t')$**:** At time $t$ the agent has the potential intention to do subaction $b$ of action $a$ using recipe $r$ at time $t'$.

$Int(t, b, a, r, t')$**:** At time $t$ the agent has the intention to do subaction $b$ of action $a$ using recipe $r$ at time $t'$.

$Ref(t, b, a, r, t')$**:** At time $t$ the agent is refrained from intending to do subaction $b$ of action $a$ using recipe $r$ at time $t'$.

We assume that the recipes for an action form a list which is represented by the predicate $NextRec$ as follows:

$NextRec(a, r, r')$**:** In the list of recipes for $a$, $r$ is followed immediately by $r'$.

Our convention is that the first recipe $r$ is indicated by writing $NextRec(a, \_, r)$ and the last recipe $r$ by $NextRec(a, r, \_)$ (because $\_$ is used as the empty recipe).

When a recipe fails, the agent needs to try another recipe. For this purpose we have the following predicate.

$FailedRec(t, a, r, t')$**:** At time $t$ the recipe $r$ for $a$ to be done at time $t'$ failed.

When the last recipe for a subaction fails, the agent needs to try another recipe for the action. In order to allow the agent to work its way up the tree of recipes, we have the following predicate.

$Parent(a, r, t, a', r', t')$**:** The subaction $a$ of action $a'$ is to be done using recipe $r$ at time $t$, while $a'$ is to be done using recipe $r'$ at time $t'$.

It may turn out that the agent tried all the recipes and cannot do an action it was asked to do. At this point it should report the failure of the whole process. This is done by using the following:

$StopPlan(t, a, t')$**:** At time $t$ the agent stops planning to do $a$ at time $t'$.

### 4.1.2 The Axioms

As we did in Section 3.1 we list all the axioms with explanations. Some of the axioms are new versions of previous axioms; we indicate this by adding $MR$ to the previous designation.

**A1-MR. Asked To Do Becomes Potential Intention**

We do the version where the single agent is initially asked to do the action. The difference is the addition of $Parent$ in the consequent.

$$ATD(t, a, t') \ \& \ t{+}1 < t' \rightarrow PotInt(t{+}1, a, \_, \_, t') \ \& \ Parent(a, \_, t', \_, \_, \_)$$

**A2-MR. Inheritance of Potential Intention** **A2.1-MR: self** The difference here is that the potential intention is inherited with the same recipe.

$$PotInt(t, b, a, r, t') \ \& \ \neg Ref(t, b, a, r, t') \ \& \ t{+}1 < t' \rightarrow PotInt(t{+}1, b, a, r, t')$$

**A2.2-MR: child** The difference here is substantial. The second conjunct in the antecedent indicates that this is a newly obtained potential intention. The $NextRec$ predicate identifies $r'$ as the first recipe. As in A1-MR the $Parent$ predicate is set correctly in the consequent.

$$PotInt(t, b, a, r, t') \ \& \ \neg PotInt(t-1, b, a, r, t') \ \& \ Bel(t, "CanDo(t', b)") \ \&$$
$$NextRec(b, \_, r') \ \& \ Mem(r', c, i, j, k) \ \& \ t+1 < t'$$
$$\rightarrow PotInt(t+1, c, b, r', t'+j) \ \& \ Parent(c, r', t'+j, b, r, t')$$

**A3-MR. Potential Intention Becomes Intention**

The difference here is that the agent has the intention to do all the subactions of some recipe for $b$.

$$PotInt(t, b, a, r, t') \ \& \ Bel(t, "CanDo(t', b)") \ \& \ t+1 < t' \ \&$$
$$\exists r', r''(NextRec(b, r', r'') \ \& \ (\forall c, i, j, k \ Mem(r', c, i, j, k) \rightarrow Int(t, c, b, r', t'+j))$$
$$\rightarrow Int(t+1, b, a, r, t')$$

**D1-MR. Chosen Recipe Fails**

The chosen recipe has a subaction that the agent does not believe it can do, hence the recipe fails. The $Parent$ predicate is needed in the antecedent for the starting time of $a$.

$$PotInt(t, b, a, r, t') \ \& \ \neg Bel(t, "CanDo(t', b)") \ \& \ Parent(b, r, t', a, r", t")$$
$$\rightarrow FailedRec(t+1, a, r, t")$$

**D2-MR. Potential Intention Inherited with Different Recipe**

If a recipe, that is not the last recipe of an action, fails, then the agent inherits the potential intention for the action, but with the next recipe for the action.

$$Parent(a', r'', t', a'', r''', t") \ \& \ FailedRec(t, a', r, t') \ \& \ NextRec(a', r, r') \ \&$$
$$r' \neq \_ \ \& \ Mem(r', b, i, j, k) \ \& \ t+1 < t'$$
$$\rightarrow \quad PotInt(t+1, b, a', r', t'+j) \ \& \ Parent(b, r', t'+j, a', r'', t')$$

**D3-MR. Recipe Failure Induces Refrain**

If a recipe fails, then a refrain is induced for each subaction of the recipe. This way the potential intentions will not be inherited via A2-MR.

$$FailedRec(t, a, r, t') \ \& \ Mem(r, b, i, j, k) \ \& \ t{+}1 < t' \rightarrow Ref(t{+}1, b, a, r, t'{+}j)$$

**D4-MR. Failure of the Last Recipe for an Action**

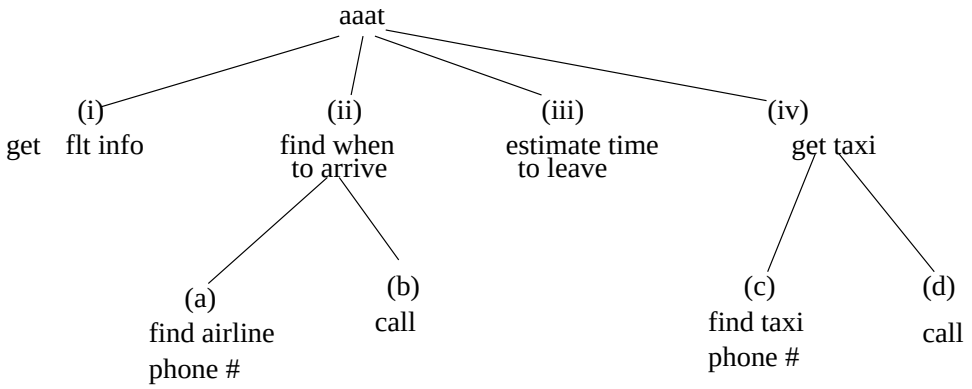


Figure 3: Additional recipe tree.

**D4.1-MR: non-root node** If the last recipe for an action fails, that failure propagates upward in the recipe tree to the parent action. The condition $a' \neq \_$ is needed in the antecedent because otherwise $a$ is the root action and has no parent.

$$PotInt(t, a, a', r", t'') \;\&\; FailedRec(t, a, r, t') \;\&\; NextRec(a, r, \_) \;\&\; Parent(a, r'', t', a', r''', t'') \;\&\; a' \neq \_ \rightarrow FailedRec(t+1, a', r'', t'')$$

**D4.2-MR: root node** If the last recipe for the root action fails, the agent stops planning the action entirely.

$$PotInt(t, a, \_, \_, t') \;\&\; FailedRec(t, a, r, t') \;\&\; NextRec(a, r, \_) \;\&\; t+1 < t' \rightarrow StopPlan(t+1, a, t')$$

### 4.1.3 Example

In this subsection we illustrate some of the axioms by using the example of Figure 2 with an additional recipe. To make things simple we again assume that there is only one recipe for $aaat$, as given there, and one recipe for $get\_taxi$, as given there. However, now there is a second recipe for $find\_when\_to\_arrive$, as shown in Figure 3. Thus there are two ways for an agent to find out when to arrive at the airport: first, it can find the url of the airline and then do a web search, or, second, it can find the airline phone number and then call. In this illustration we concentrate on intention formation with these two recipes.

We also assume that there is only one agent, $G$, whose name will not be written in the formulas in accordance with the convention for single agents. Again, we start with the $ATD$ statement, now written as $ATD(10, aaat, 50)$, that is, at time 10 the agent was asked to do $aaat$ at time 50.

- Axiom A1-MR entails two facts: $PotInt(11, aaat, \_, \_, 50)$ and $Parent(aaat, \_, 50, \_, \_, \_)$.

- A2.1-MR entails $PotInt(12, aaat, \_, \_, 50)$.

Assume that the following statements are known:
$Bel(11, "CanDo(50, aaat)")$,
$NextRec(aaat, \_, r_0)$, and
$Mem(r_0, find\_when\_to\_arrive, 2, 3, 7)$,
that is, the agent believes that it can do $aaat$, the first recipe for $aaat$ is $r_0$, and the second member of this recipe is $find\_when\_to\_arrive$, starting at relative time 3 and ending at relative time 7.

- A2.2-MR entails $PotInt(12, find\_when\_to\_arrive, aaat, r_0, 53)$ and $Parent(find\_when\_to\_arrive, r_0, 53, aaat, \_, 50)$.

Recall that we have two recipes for $find\_when\_to\_arrive$. The first recipe uses the web, the second the phone. We write these as $r_1$ and $r_2$ respectively. Let us assume the following four statements dealing with the action $find\_when\_to\_arrive$:
$Bel(12, "CanDo(52, find\_when\_to\_arrive)")$,
$NextRec(find\_when\_to\_arrive, \_, r_1)$,
$Mem(r_1, find\_url, 1, 1, 2)$, and
$Mem(r_1, searchweb, 2, 3, 4)$.

- A2.2-MR entails the following four statements:
  $PotInt(13, find\_url, find\_when\_to\_arrive, r_1, 54)$,
  $Parent(find\_url, r_1, 54, find\_when\_to\_arrive, r_0, 53)$
  $PotInt(13, searchweb, find\_when\_to\_arrive, r_1, 56)$,
  $Parent(searchweb, r_1, 56, find\_when\_to\_arrive, r_0, 53)$.

Next suppose that $Bel(13, "CanDo(56, searchweb)")$ does not hold. This is the case where the chosen recipe, $r_1$ does not work. Hence

- D1-MR entails $FailedRec(14, find\_when\_to\_arrive, r_1, 53)$.

Now, using the statements
$NextRec(find\_when\_to\_arrive, r_1, r_2)$,
$Parent(find\_when\_to\_arrive, r_1, 53, aaat, r_0, 50)$,
$Mem(r_2, find\_airlinephone, 1, 1, 2)$ and
$Mem(r_2, call\_airline, 2, 3, 4)$

- D2-MR entails $PotInt(15, find\_airlinephone, find\_when\_to\_arrive, r_2, 54)$ and
  $PotInt(15, call\_airline, find\_when\_to\_arrive, r_2, 56)$

as well as some instances of the $Parent$ predicate, not needed for this illustration, and the process continues with this recipe.

Note also that

- D3-MR entails
  $Ref(15, find\_url, find\_when\_to\_arrive, r_1, 54)$ and
  $Ref(15, searchweb, find\_when\_to\_arrive, r_1, 56)$.

So some potential intentions will not be inherited.
Assuming that recipe $r_2$ is appropriate, then,

- A2.2-MR and A3-MR entail
  $Int(17, find\_when\_to\_arrive, aaat, r_2, 55)$

and if intentions are obtained for all the other members of $r_0$

- $Int(18, aaat, \_, r_0, 50)$ will follow.

#### 4.1.4 Meta-theorem

Theorem 1, given for the case of a single recipe can be extended to the theory of the multi-recipe case. Thus this theory also has an intended model and is consistent. Furthermore, it is possible to extend Theorem 2 to the case of multiple recipes. Before doing so, we find it useful here to think of a recipe tree in an alternative manner. At the beginning of Section 4 we indicated that in the case of multiple recipes the recipe tree becomes a complex and-or tree. But here we think of a recipe tree as a regular tree, just as we used the concept in the case of single recipe actions. Then with each action we associate a set of recipe trees, where each recipe tree is obtained by choosing a specific recipe for each complex action. In other words, a recipe tree is obtained from the and-or tree by choosing one subtree for each "or" branch.

The generalization of Theorem 2 is as follows:

**Theorem 4** *Suppose an agent is asked at time $t$ to do an action $a$ starting at time $t+s$, there are $w$ recipe trees, the maximum height of all recipe trees is $h$, the maximum time for any recipe is $k$ units of time, and $s \geq 2hw + 3$. Further, suppose there is a recipe tree for which the agent believes that it can do each subaction (at the needed time) itself, is not refrained at any time from doing the action or any subaction, and that it in fact can do them. Then the agent will complete the task at time $t + s + k$. Moreover, in the worst case, any value for $s$ smaller than $2hw + 3$ will not, in general, be sufficient to start the action at time $t + s$, and hence finish at time $t + s + k$.*

**Proof :** As in Theorem 2 we show that intention formation takes at most $2hw + 2$ units of time by showing abbreviated versions of the key formulas that become true as time changes. However, we only go to the point where the second recipe tree is started. To consider the worst case we assume that all the $w$ recipe trees are entirely different and have height $h$.

**t** $ATD(a)$

**t+1** $PotInt(a, \_)$ $a$ is not a child node for a recipe

**t+2** $PotInt(b1, r1)$, $PotInt(b2, r1)$, ... the children node of $a$ in recipe $r1$

...

**t+h+1** $PotInt(z1, r1^{h1})$, $PotInt(z2, r1^{h2})$, ... the nodes at the bottom level for the first recipe tree

**t+h+2** $FailedRec(r1^{hk})$ a recipe fails at the next to bottom level

...

**t+2h+1** $FailedRec(a, r1)$ the first recipe tree fails

**t+2h+2** $PotInt(b1', r2)$, $PotInt(b2', r2)$, ... reasoning continues with the next recipe

...

Thus it takes 1 unit of time to get the initial potential intention for the root node and $2h$ units to recognize the failure of the first recipe tree (in the worst case). Each recipe tree may take up to $2h$ units including the $w^{th}$ (last) one, where instead of getting $FailedRec$ going back up the tree, $Int$ will be entailed. For this last recipe tree one more time unit is needed to get $Int(a)$. Hence, altogether, the process may take up to $1 + 2hw + 1 = 2hw + 2$ units.

∎

In the example for $aaat$, using $h = 2$ and $w = 2$, we obtain $2hw + 2 = 10$, hence intention formation takes at most 10 units of time. Note also that the formula for $s = 2hw + 3$ reduces to $2h + 3$ in case $w = 1$, which is the formula of Theorem 2.

## 4.2 Multiple agents

We only sketch the case of multiple agents and multiple recipes here. It was observed in the previous section that in the case of multiple recipes a backtracking mechanism must be introduced into the meta-language. We accomplished this by placing the recipes for each action into a list using the $NextRec$ predicate, explicitly stating that a recipe failed by using the $FailedRec$ predicate, and keeping track of where the agent's reasoning is in the complex and-or tree of recipes by using the $Parent$ predicate.

We begin by considering the case of multiple agents with single recipes in a more general way. In our work in Section 3 we assumed that an agent, say $Agent_0$, subcontracts an action $a$ to the first agent, say $Agent_1$, that it believes can do $a$. We did not deal with the case where $Agent_1$ itself does not believe that it can do $a$. In this case it would be reasonable for $Agent_0$ to try another agent, say $Agent_2$, that $Agent_0$ believes can do $a$. Setting up this process involves backtracking in a way that is similar to the case of multiple recipes. Namely, each agent will have a list of agents; if the chosen agent does not believe that it can do the action, it will fail and the next agent will be chosen. Hence an agent backtracking mechanism must be set up in a way that is similar to the recipe backtracking mechanism.

Consider now the case of multiple agents and multiple recipes where during the reasoning process there may be both failed recipes and failed agents. In this case a double backtracking must be set up. A natural way to do this is to nest one backtracking inside the other one. For example, using the recipes for the outer backtracking would mean

that the agent will start with the first recipe always, but then it may have to subcontract to several agents before finding one that believes it can do the relevant action. Each subcontracting agent, in turn, will start with the first recipe and try subcontracting to other agents, trying the next recipe only if none of the agents believes that it can do the subcontracting action.

We now sketch a scenario for the example given in Figure 3. Recall that there are two recipes for $find\_when\_to\_arrive$. In the previous subsection we had only one agent, but let's go back to the case of two agents $G$ and $H$ from Section 1.3. Suppose also that when $G$ tries to subcontract the $searchweb$ action to $H$, $H$ does not believe that it can do it. In that case, since there are no other agents, the first recipe for $find\_when\_to\_arrive$ will fail and the second recipe must be tried. Since H finds phone numbers, the action $find\_when\_to\_arrive$ will be subcontracted to $H$ which may then subcontract the $call\_airline$ action back to $G$.

Our goal now is to generalize Theorem 2 to the case of multiple agents and multiple recipe trees. We start with the case where there is only one recipe tree and at least two agents, that is, $w = 1$ and $z \geq 2$.

**Theorem 5** *Suppose an agent is asked at time $t$ to do an action $a$ starting at time $t+s$, there is one recipe tree, the height of the recipe tree is $h$, the maximum time for any recipe is $k$ units of time, there are $z$ agents, $z \geq 2$, and $s \geq (h+1)(3z-1)+1$. Further, suppose the agent believes that it can do each subaction (at the needed time) itself or can successfully subcontract it to some agent that it believes can do it (at the needed time), and in fact there is always such an agent, and neither agent is refrained at any time from doing the action or any subaction, and that it in fact can do it. Then the agent will complete the task at time $t+s+k$. Moreover, any value for $s$ smaller than $(h+1)(3z-1)+1$ will not, in general, be sufficient to start the action at time $t+s$.*

**Proof :** As in Theorem 3 we show a portion of the intention formation process in the worst case using abbreviated versions of the key formulas. Agents are represented by the numbers 1,...z.

**t** $ATD(1, a)$ the first agent is asked to do $a$

**t+1** $PotInt(1, a)$

**t+2** $FailedAgent(1, a)$ the first agent does not believe that it can do $a$

**t+3** $ATD(2, a)$ the second agent is asked to do $a$

**t+4** $PotInt(2, a)$

**t+5** $FailedAgent(2, a)$ the second agent does not believe that it can do $a$

**...**

**t+2+3(z-2)** $FailedAgent(z-1, a)$ the next-to-last agent does not believe that it can do $a$

**t+2+3(z-2)+1** $ATD(z, a)$ the last agent is asked to do $a$

**t+2+3(z-2)+2** $PotInt(z, a)$ the last agent believes it can do $a$

**t+2+3(z-2)+3** $PotInt(z, b)$, $PotInt(z, c)$, ... where $b$ and $c$ are the children nodes of $a$

**...**

The calculation of the time periods is as follows (in the worst case). At each level the first agent is asked and fails: that is 2 units. The next $z-2$ agents take $3(z-2)$ units, and the last agent takes 2 time units. Altogether this is $3(z-2)+4 = 3z-2$ units. This process is repeated for the $h+1$ levels, taking $(h+1)(3z-2)$ time units to reach and succeed at the bottom level. To this must be added $h+1$ units, going back up the tree with the intentions. Thus the reasoning takes $(h+1)(3z-1)$ time units.

∎

In the example for $aaat$ assuming a single recipe tree and two agents, that is, $z = 2$ and $h = 2$, we obtain $(h+1)(3z-1)$ = 15 units that should be allocated for reasoning.

We end with the most general result for success where there are multiple recipe trees and multiple agents.

**Theorem 6** *Suppose an agent is asked at time $t$ to do an action $a$ starting at time $t+s$, there are $w$ recipe trees, the maximum height of a recipe tree is $h$, the maximum time for any recipe is $k$ units of time, there are $z$ agents, $z \geq 2$, and $s \geq (h+1)(3z-1)w + w$. Further, suppose for some recipe tree the agent believes that it can do each subaction (at the needed time) itself or can successfully subcontract it to some agent that it believes can do it (at the needed time), and in fact there is always such an agent, and neither agent is refrained at any time from doing the action or any subaction, and that it in fact can do it. Then the agent will complete the task at time $t+s+k$. Moreover, any value for $s$ smaller than $(h+1)(3z-1)w+w$ will not, in general, be sufficient to start the action at time $t+s$.*

**Proof :** We just sketch here the calculation using the proof of Theorem 4. We start with the same process of multiple agents for the first recipe tree. This takes $(h+1)(3z-1)$ units. This process is done for all the $w$ recipe trees, adding one time unit to get to each new recipe tree, finally obtaining $(h+1)(3z-1)w + (w-1)$ time units for the reasoning process.

In the case of $aaat$ with 2 agents and two recipe trees, the intention formation process will take at most $3 \times 5 \times 2 + 1 = 31$ time units.

# 5 Related Work

Intentions in the context of SharedPlans were studied in [19, 20], but no semantics were given. Our starting point in this paper was the axioms presented by Grosz and Kraus but our requirements for an agent having an intention are much stronger than those presented in [19] where an agent may have an intention also when having a partial

plan. For example, the agent may have only partial knowledge about a recipe, but a plan how to complete it; it may have only potential intentions toward subactions. On the other hand, we require that in order for the agent to have an intention, it must have a full detailed plan to do the action and that it has adopted the appropriate intentions and beliefs. These requirements together with our semantics enable us to state and prove various properties and theorems about agent intentions and actions.

Since the definition of $Int$ in our model is much stronger than the $Int.To$ of SharedPlans, the $PotInt$ predicate of our model plays a more important role in the agent's reasoning than the $Pot.Int.To$ does in [19]. In [19] $Pot.Int.To$ is used only as an intermediate operator until $Int.To$ is adopted. In our model the $PotInt$ is kept for the duration of the agent's need for the associated intention and is used during the intention formation process and for continuous verification of the minimal requirements for having the intention.

The SharedPlan model of collaborative planning uses the mental state model of plans [42]. Bratman [5] also argues for a mental-state view of plans, emphasizing the importance of intentions to plans. He argues that intentions to do an action play three roles in rational action: having an intention to do an action constrains the other intentions an agent may adopt, focuses means-ends reasoning, and guides replanning. These roles are even more important for collaborative activity than for individual activity. In our model $Int$ and $PotInt$ play these roles.

Most of the models of intention of a single agent do not have context parameters (e.g., [44, 51]). However, once agents are working in a group, such parameters are very important to enhance cooperation [20]. Since we apply a syntactic approach, the introduction of such parameters is not difficult compared with adding such parameters in modal logics [25]. Thus, one of the parameters for both $PotInt$ and $Int$ is the agent that is assisted by the performance of the intended action. Another parameter is the action in whose context the intended action is performed. In [19], an intention has a general parameter of context which includes the intentional context in which the intended action is being performed (among other things). Since subcontracting is the main focus of our paper, and the intentional context is very important for the interactions between the agents, our intentions have these two explicit parameters. These parameters are different from relativized intentions such as in Cohen and Levesque [8]. They allow an "escape clause" or "background condition" parameter. Once this clause becomes false, the agent drops the associated intention. In our case, explicit communication from the assisted agent may lead to dropping the intention, as discussed above.

Castelfranchi [7] studies the notion of intention for describing and understanding the activities of groups and organizations in the context of improving the exchange between AI and social and management approaches to cooperative work. His motivation is different from our aim of developing a formal logic of beliefs and intentions.

Others proposed models for team intentions (e.g., [32, 9, 50, 52]) while we focus on the intentions of individual agents that may subcontract to other cooperative agents.

There were several attempts to develop possible worlds semantics for agents' intentions [43, 25, 15, 8, 28, 29, 2, 55]. Some problems arise with these attempts such as that in most of them intentions are closed under Modus Ponens or under logical equivalence and that the relations between the action's recipe and the intention are not well defined. Using a syntactic approach provides more freedom in modeling the way

agents' intentions and beliefs change over time. See [55] for an excellent survey. We discuss here a few models that address some of the problems that we consider.

Konolige and Pollack [25] use a special type of possible world models which they refer to as cognitive structures. It is equivalent to Chellas' minimal model semantics (that is different from our first-order minimal models). Using the cognitive structures, Konolige and Pollack eliminate the closed under inferences problem with respect to intentions. However, their intentions are still closed under logical equivalence. In order to capture relations between intentions, they introduce the concept of an embedding graph among intentions. This extension allows them to model a static relationship between intention and beliefs, but it is not appropriate for modeling a team of agents working together dynamically to plan and perform a complex action.

Georgeff and Rao [15] consider the problem of intention maintenance in the context of changing beliefs and desires. They extend the standard possible-world semantic logics by introducing forms for *only* modalities of belief, desire, and intention along the lines of Levesque's only belief operator [31]. Intuitively, when moving from one time to another they maintain as many old intentions as possible that satisfy some constraints of what is a consistent set of intentions. They specify semantic constraints on their models that reflect this intuition (but do not prove completeness). Intention maintenance is done in our model via the inheritance axioms. We distinguish between $PotInt$ and $Int$: $PotInt$ is always inherited unless the agent is explicitly refrained from doing the action. An intention to do an action is inherited only if the associated $PotInt$ is inherited, the agent continues to believe that it can do the action, and if each subaction (if exists) in the recipe of the action is either intended by the agent or an assisting agent.

Sonenberg and her colleagues [45, 52, 23] present a very comprehensive and interesting possible-world semantics model for beliefs, intentions, and time, and propose the concepts of a plan graph and a plan structure. We, on the other hand, propose a syntactic approach for beliefs and intentions and time, and explicitly express them in our logic. They use these concepts for defining joint intentions and for reasoning about team activity while we focus on an agent's individual intention and its capability to subcontract some actions to other agents. Although their model enables the expression of much more complex plans than ours, the lack of explicit terms for expressing time limits their ability to reason about the ways agents' intentions change over time. We also model explicitly the process of backtracking in case of a failure.

Singh and Asher [51] present a formal theory of intentions and beliefs based on Discourse Representation Theory. As in our model, this theory does not assume that agents are perfect reasoners. However, they do not model the formation of intentions and its relation to planning.

Cohen and Levesque [8, 10] have a notion of intention based on persistence goals (P-GOAL). They assume that if an agent has a P-GOAL toward a proposition, then the agent believes that this proposition is not true now, but that it will be true at some time in the future. The agent will drop a persistent goal $p$ only if it comes to believe that $p$ is true or that $p$ is impossible. In their logic, time doesn't explicitly appear in the proposition; thus, they cannot express a P-GOAL toward propositions that will be true at some specific time in the future or consider situations where a proposition is true now, but which the agent believes will become false later and therefore has to make

a P-GOAL true again after it becomes false. Our intentions are toward actions. Since time is explicit in our logic, we can express intentions toward performing a given action at a specific time, in addition to expressing Cohen and Levesque's attitude P-GOAL. This can be done by defining in our language predicates similar to those of Cohen and Levesque (e.g., BEL, GOAL, P-GOAL) and adding to our theorem appropriate axioms that characterize these predicates.

Sadek [46] who extended and refined Cohen and Levesque's theory of intentions introduced the concept of *need* or *potential intention*. In his model, the concept of need is logically characterized from choice, belief and intention: an agent needs $\phi$ if the fact that it does not believe $\phi$ is a sufficient condition for intending to believe $\phi$. In our model potential intention is a basic concept that means roughly that the agent has been given a task and is determining whether it can do it (perhaps with help).

Goldman and Lang [17] also extended Cohen and Levesque's work. They use Allen's temporal logic [1] as a foundation and also use syntactic models of belief. They are able to formalize complex intentional actions, particularly with deadlines. Intentions are motivated by goals, but the process of intention foundation is not modelled, in particular, subcontracting and communications between agents are not considered.

SRI's Procedural Reasoning System (PRS) [39] is a framework for constructing real-time reasoning systems. Some of our concepts are similar to features of PRS. For example, a PRS Act describes the steps of a procedure and hence is something like a simple recipe in our terminology that consists of basic level actions. A PRS intention involves Acts and Subacts and is a kind of combination of our intention along with a recipe. In PRS time is not represented explicitly and there are important differences with our approach concerning subcontracting and other concepts.

Wooldridge and Jennings [56] provide a formal model of the cooperative problem solving process for multi-agent systems using a logical formalism that combines aspects of modal, temporal, and dynamic logic. Dunin-Keplicz and Verbrugge [54] investigate the notion of collective intention in the context of cooperative problem solving for multiagent systems. They formalize collective intentions in a multi-modal logical framework and prove the completeness of the logic with respect to a class of Kripke models.

An intention in our model is toward an action and not toward a proposition as in most of the models mentioned above. This somewhat limits intention, but it allows us to clearly relate intentions and recipes. Our intentions can be converted to proposition by using an operator like "Do". The other models' intentions can be converted to actions by using a special action "Make-True".

An interesting dual treatment of agents that, like ours, has both an agent language ("first-person account") [16] and a meta language ("third-person account") [30], uses the Golog family of languages based on the situation-calculus. Those papers (unlike our own) are more focussed on knowledge conditions than on intentions, and also do not take time-taken-to-plan into account.

Wooldridge and his colleagues [48, 57] are concerned with intention formation and intention reconsideration. They focus on the issue of designing agents that can balance the amount of time spent in reconsidering their intentions against the amount of time spent acting to achieve them. They use a formal but non-logical approach and it is an open question how the balance between intention reconsideration and acting could be

modeled in our logic.

Subcontracting was considered mainly in situations of self-interested agents where the emphasis was on finding mechanisms for motivating the subcontractor to perform its task as required (see a survey in [26].) In this paper we assume that the agents are cooperative and willing to perform subcontracted actions if they can.

Since we followed the mental state approach for planning and focused on subcontracting, we adapt a rather simple approach for reasoning about actions. There is a very rich literature on reasoning about actions. For example, Konolige [24] formalizes reasoning about the knowledge, belief, and actions of agents. He uses an object language to describe agent beliefs and a metalanguage to study the object language. Actions are described in the situation calculus. But the intentions of agents are not considered. And although it is syntactic (like ours), Konolige's approach explicitly assumes closure under inference. While the inference rules do not have to be traditional (strictly deductive) ones, nevertheless all the inferential conclusions (theorems) resulting from them are assumed known or believed by the agent — they do not come in little by little over time as the agent manages to prove them one by one — and so this is clearly not realistic. Our approach by contrast employs an explicit time mechanism to track the evolution of an agent's gradual process of inference.

# 6 Summary and Future Work

We presented a formal logical calculus that can be regarded as a meta-logic that describes the reasoning and activities of agents. The explicit representation of evolving time is an important feature of this approach. We dealt with the case where agents are assigned tasks for which a recipe is known. Recipes have a tree structure. An agent may subcontract some of the actions/subactions to other agents. Our emphasis is on developing a framework that models the beliefs, intentions, and actions of agents as they change over time. We present a syntactic approach, propose a minimal model semantics and prove that the meta-theory is consistent and has a minimal model. The true statements in the minimal model associated with the agents' mental states and actions characterize the agents that are described by the meta-logic. Using this semantics, rather than possible world semantics, allows us to model agents activity more realistically and to prove several results to show that under the appropriate conditions the agents will act as desired.

We plan to extend this work in several ways. At present we have results only for strongly positive (agents always successfully subcontract actions/subactions, their beliefs about their activities are correct, and communication always succeeds) and strongly negative (there is a subaction that no agent can do) cases. We will consider more complex situations. Additionally we will deal with situations where agents have SharedPlans (and not only subcontract actions).

# References


[1] James Allen. Towards a general theory of action and time. *Artificial Intelligence*, 23(2):123–144, 1984.

[2] E. Alonso. A formal framework for the representation of negotiation protocols, 1997.

[3] W. Bibel. *Automated Theorem Proving*. Friedr. Vieweg and Sohn Verlagsgesellschaft GmbH, Braunschweig, 1982.

[4] W.W. Bledsoe and D.W. Loveland. *Automated theorem proving*. American Mathematical Society, Providence, 1984.

[5] M. E. Bratman. *Intention, Plans, and Practical Reason*. Harvard University Press, Cambridge, MA, 1987.

[6] A. Bundy. *The computer modeling of mathematical reasoning*. Academic Press, London, 1983.

[7] C. Castelfranchi. Commitments: From individual intentions to groups and organizations. In *ICMAS 95*, 1995.

[8] P. Cohen and H. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42:263–310, 1990.

[9] P. Cohen and H. Levesque. Teamwork. *Noûs*, pages 487–512, 1991.

[10] P. R. Cohen and H. Levesque. Rational interaction as the basis for communication. In Philip R. Cohen, Jerry L. Morgan, and Martha E. Pollack, editors, *Intentions in Communication*, pages 221–256. MIT Press, Cambridge, MA, 1990.

[11] P. R. Cohen, J. Morgan, and M. E. Pollack (editors). *Intentions in Communication*. MIT Press, 1990.

[12] E. Davis. *Representation of Commonsense Knowledge*. Morgan Kaufman Publishers, Inc, California, 1990.

[13] J. de Rivières and H. Levesque. The consistency of syntactical treatments of knowledge (how to compile quantificational modal logics into classical FOL. *Computational Intelligence*, 4:31–41, 1988.

[14] J. Doyle, Y. Shoham, and M. Wellman. A logic of relative desire. In *Proc. of the 6th International Symposium on Methodologies for Intelligent Systems*, 1991.

[15] M. Georgeff and A. Rao. The semantics on intention maintenance for rational agents. In *Proc. of IJCAI95*, pages 704–710, Montreal, Canada, 1995.

[16] G. De Giacomo, Y. Lesperance, H.J. Levesque, and S. Sardina. On the semantics of deliberation in indigolog—from theory to implementation. In *KR'02*, 2002.

[17] R. P. Goldman and R. R. Lang. Intentions in time. Technical Report TUTR 93–101, Tulane University, 1993.

[18] J. Grant, S. Kraus, and D. Perlis. A logic for characterizing multiple bounded agents. *Autonomous Agents and Multi-Agent Systems Journal*, 3(4):351–387, 2000.

[19] B. J. Grosz and S. Kraus. Collaborative plans for complex group activities. *Artificial Intelligence Journal*, 86(2):269–357, 1996.

[20] B. J. Grosz and S. Kraus. The evolution of sharedplans. In A. Rao and M. Wooldridge, editors, *Foundations and Theories of Rational Agency*, pages 227–262. Kluwer Academic Publishers, 1999.

[21] Barbara Grosz and Candace Sidner. A reply to Hobbs. In P. Cohen, J. Morgan, and M. Pollack, editors, *Intentions in Communication*, pages 461–462. Bradford Books/MIT Press, Cambridge, MA, 1990.

[22] A. Haass. The syntactic theory of belief and knowledge. *Artificial Intelligence*, 28(3):245–293, 1983.

[23] D. Kinny, M. Ljungberg, A. S. Rao, E. Sonenberg, G. Tidhar, and E. Werner. Planned team activity. In C. Castelfranchi and E. Werner, editors, *Artificial Social Systems, Lecture Notes in Artificial Intelligence (LNAI-830)*, Amsterdam, The Netherlands, 1994. Springer Verlag.

[24] K. Konolige. A first-order formalisation of knowledge and action for a multiagent planning system. *Machine Intelligence*, 10:41–72, 1982.

[25] K. Konolige and M. E. Pollack. A representationalist theory of intention. In *Proc. of IJCAI-93*, pages 390–395, Chambery, France, August 1993.

[26] S. Kraus. An overview of incentive contracting. *Artificial Intelligence Journal*, 83(2):297–346, 1996.

[27] S. Kraus. *Strategic Negotiation in Multiagent Environments*. MIT Press, Cambridge, USA, 2001.

[28] S. Kraus, K. Sycara, and A. Evenchik. Reaching agreements through argumentation: a logical model and implementation. *Artificial Intelligence*, 104(1-2):1–69, 1998.

[29] S. Kumar, M. J. Huber, D.R. McGee, P.R. Cohen, and H.J. Levesque. Semantics of agent communication languages for group interaction. In *AAAI 2000*, pages 42–47, 2000.

[30] Y. Lesperance. On the epistemic feasibility of plans in multiagent systems specifications. In *ATAL'01*, 2001.

[31] H. Levesque. All I know: A study in autoepistemic logic. *Artificial Intelligence*, 42:263–309, 1990.

[32] H. Levesque, P. Cohen, and J. Nunes. On acting together. In *Proceedings of AAAI-90*, pages 94–99, Boston, MA, July 1990.

[33] Karen Lochbaum, Barbara Grosz, and Candace Sidner. Models of plans to support communication: An initial report. In *Proceedings of the 8th National Conference on Artificial Intelligence (AAAI-90)*, pages 485–490, Cambridge, MA, 1990. MIT Press.

[34] R Montague. Syntactical treatments of modality, with corollaries on reflection principles and finite axiomatizability. In *Modal and Many-Valued Logics (Acta Philosophica Fennica, vol. 16)*. Academic Bookstore, Helsinki, 1963. Reprinted in R. Montague (1974). *Formal Philosophy*, New Haven, pp. 286-302.

[35] R. Moore. A formal theory of knowledge and action. In J. Hobbs and R. Moore, editors, *Formal Theories of the Commonsesnse World*. ABLEX publishing, Norwood, N.J., 1985.

[36] L. Morgenstern. *Foundations of a Logic of Knowledge, Action, and Communication*. PhD thesis, New York University, 1988.

[37] L. Morgenstern. Inheritance comes of age: Applying nonmonotonic techniques to problems in industry. In *IJCAI*, pages 1613–1621, 1997.

[38] M. Morreau and S. Kraus. Syntactical treatments of propositional attitudes. *Artificial Intelligence Journal*, 106:161–177, 1998.

[39] K. L. Myers. User guide for the procedural reasoning system. Technical report, Artificial Intelligence Center, SRI International, 1997.

[40] D. Perlis. Language with self references I: Foundation. *Artificial Intelligence*, 25:301–322, 1985.

[41] R. Perrault. An application of default logic to speech act theory. In P.R. Cohen, J.L. Morgan, and M.E. Pollack, editors, *Intentions in Communication*, pages 161–185. Bradford Books at MIT Press, 1990.

[42] Martha E. Pollack. Plans as complex mental attitudes. In P.N. Cohen, J.L. Morgan, and M.E. Pollack, editors, *Intentions in Communication*. Bradford Books, MIT Press, 1990.

[43] A. Rao and M. Georgeff. Deliberation and its role in the formation of intention. In *Proceedings of the Seventh Conference on Uncertainty in Artificial Intelligence*, San Mateo, California, 1991. Morgan Kaufmann Publishers, Inc.

[44] A. Rao and M. Georgeff. Modeling rational agents within BDI architecture. In *Proc. of the Second International Conference of Knowledge Representation*, pages 473–484, San Mateo, 1991. Morgan Kaufman Publishers.

[45] A. Rao, M. P. Georgeff, and E. A Sonenberg. Social plans: A preliminary report. In *Decentralized Artificial Intelligence, Volume 3*, pages 57–76. Elsevier Science Publishers, 1992.

[46] M. D. Sadek. A study in the logic of intention. In *Proceedings of KR*, pages 462–473, 1992.

[47] E. Sandewall. *Features and Fluents*. Oxford University Press, 1994.

[48] M. Schut, M. Wooldridge, and S. Parsons. Reasoning about intentions in uncertain domains. In *Proceedings of the Sixth European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU-2001)*, Toulouse, France, September 2001.

[49] M. Shanahan. *Solving the Frame Problem*. The MIT Press, 1997.

[50] M. P. Singh. The intentions of teams: Team structure, endodeixis, and exodeixis. In *Proceedings of the 13th European Conference on Artificial Intelligence (ECAI)*, pages 303–307. Wiley, 1998.

[51] M. P. Singh and N. M. Asher. A logic of intentions and beliefs. *Journal of Philosophical Logic*, 22:513–544, 1993.

[52] E. Sonenberg, G. Tidhar, E. Werner, D. Kinny, M. Ljungberg, and A. Rao. Planned team activity. Technical Report 26, Australian Artificial Intelligence Institute, Australia, 1992.

[53] R. Thomason. A note on syntactical treatments of modality. *Synthese*, 44:391–395, 1980.

[54] R. Verbrugge and B. Dunin-Keplicz. Collective intentions. *Fundamenta Informatica*, 49:271–295, 2002.

[55] M. Wooldridge. *Reasoning about Rational Agents*. The MIT Press, 2000.

[56] M. Wooldridge and N. R. Jennings. The cooperative problem-solving process. *Journal of Logic and Computation*, 9(4):563–592, 1999.

[57] M. Wooldridge and S. Parsons. Intention reconsideration reconsidered. In J. P. Muller, M. Singh, and A. Rao, editors, *Intelligent Agents V Springer-Verlag Lecture Notes in AI Volume 1365*, 1999.

[58] A. Zaniolo, S. Ceri, C. Faloutsos, R. T. Snodgrass, V. S. Subrahmanian, and R. Zicari. *Advanced Database Systems*. Morgan Kaufmann Publishers, Inc., 1997.

A summary of predicates and axioms

| Name | Meaning |
|---|---|
| $ATD(t, n, m, b, a, t')$ | Agent $n$ asks agent $m$ to do action $b$ in the context of action $a$ at time $t'$. |
| $PotInt(t, m, n, b, a, t')$ | Agent $m$ directly assisting agent $n$ has the potential intention to do action $b$ in the context of action $a$ at time $t'$. |
| $Int(t, m, n, b, a, t')$ | Agent $m$ directly assisting agent $n$ has the intention to do action $b$ in the context of action $a$ at time $t'$. |
| $Ref(t, m, n, b, a, t')$ | Agent $m$ refrains agent $n$ from intending to do action $b$ in the context of action $a$ at time $t'$. |
| $BL(a, d)$ | $a$ is basic level, takes $d$ units of time to complete. |
| $Rec(a, r)$ | $r$ is the unique recipe for action $a$. |
| $Mem(r, b, i, j, k)$ | In recipe $r$, $b$ is the subaction which is the $i$'th member of the recipe starting at relative time $j$ and ending at relative time $k$. |
| $Bel(t, n, f)$ | Agent $n$ believes statement $f$. |
| $CanDo(t, n, a)$ | Agent $n$ can do action $a$. |
| $Tell(t, n, m, f)$ | Agent $n$ tells $f$ to agent $m$. |
| $Ini(t, m, n, a)$ | Agent $m$ directly assisting agent $n$ initiates action $a$. |
| $Done(t, a)$ | Action $a$ has just been done successfully. |
| $Stop(t, m, n, b, a)$ | Agent $m$ directly assisting agent $n$ is instructed to stop action $b$ in the context of action $a$. |
| $Prefer(t, n, a, b)$ | Agent $n$ prefers to do action $a$ over action $b$ |
| $Conf(t, n, a, b)$ | For agent $n$ action $a$ conflicts with action $b$. |

Table 2: A summary of predicates defined in section 2.1. In all the definitions above $t$ is the time of the proposition.

| Name | Meaning |
|---|---|
| $ATD(t, a, t')$ | The agent is asked to do $a$ at time $t'$. |
| $PotInt(t, b, a, r, t')$ | The agent has the potential intention to do subaction $b$ of action $a$ using recipe $r$ at time $t'$. |
| $Int(t, b, a, r, t')$ | The agent has the intention to do subaction $b$ of action $a$ using recipe $r$ at time $t'$. |
| $Ref(t, b, a, r, t')$ | The agent is refrained from intending to do subaction $b$ of action $a$ using recipe $r$ at time $t'$. |
| $NextRec(a, r, r')$ | In the list of recipes for $a$, $r$ is followed immediately by $r'$. |
| $FailedRec(t, a, r, t')$ | The recipe $r$ for $a$ to be done at time $t'$ failed. |
| $Parent(a, r, t, a', r', t')$: | The subaction $a$ of action $a'$ is to be done using recipe $r$, while $a'$ is to be done using recipe $r'$ at time $t'$. |
| $StopPlan(t, a, t')$ | The agent stops planning to do $a$ at time $t'$. |

Table 3: A summary of predicates defined in section 4.1.1 for intention formation with multiple recipes. In all the definitions above $t$ is the time of the proposition.

| Axiom | Title | Section |
|---|---|---|
| A1 | Asked To Do Becomes Potential Intention | 3.1.1 |
| A2 | Inheritance of Potential Intention | 3.1.1 |
| A3 | Potential Intention Becomes Intention | 3.1.1 |
| A4 | Inheritance of Refrain | 3.1.1 |
| B2 | Subcontracting an Action to an Agent | 3.1.2 |
| B3 | Potential Intention Becomes Intention (Subcontracting Version) | 3.1.2 |
| B4 | Inheritance of Refrain By an Assisting Agent | 3.1.2 |
| B5 | Communication of Assisting Agent About Refrain | 3.1.2 |
| C1 | Initiation of Requested Action | 3.1.3 |
| C2 | Inheritance of Initiate for First Subaction | 3.1.3 |
| C3 | Inheritance of Initiate for Later Subaction | 3.1.3 |
| C4 | Done for Actions | 3.1.3 |
| C5 | Action Performance Observed | 3.1.3 |
| D1 | Initiate Stop Action in case of Failure | 3.1.4 |
| D2 | Stop Action Propagated Up | 3.1.4 |
| D3 | Stop Triggers Refrain Down | 3.1.4 |
| IC1 | Preference among Conflicting Actions | 3.1.5 |
| IC2 | Consistency of Recipe Subactions | 3.1.5 |
| IC3 | Consistency of Recipe Timing | 3.1.5 |
| IC4 | Uniqueness of Recipe Information | 3.1.5 |
| A1-MR | Asked To Do Becomes Potential Intention | 4.1.2 |
| A2-MR | Inheritance of Potential Intention | 4.1.2 |
| A3-MR | Potential Intention Becomes Intention | 4.1.2 |
| D1-MR | Chosen Recipe Fails | 4.1.2 |
| D2-MR | Potential Intention Inherited with Different Recipe | 4.1.2 |
| D3-MR | Recipe Failure Induces Refrain | 4.1.2 |
| D4-MR | Failure of the Last Recipe for an Action | 4.1.2 |

Table 4: A list of axioms