Specifying Communicative Multi-Agent Systems with ConGolog

Steven Shapiro
Department of Computer Science
University of Toronto
Toronto, ON, Canada M5S 3G4
steven@ai.toronto.edu

Yves Lespérance
Department of Computer Science
Glendon College, York University
Toronto, ON, Canada M4N 3M6
lesperan@yorku.ca

Hector J. Levesque
Department of Computer Science
University of Toronto
Toronto, ON, Canada M5S 3G4
hector@ai.toronto.edu

Abstract

In this paper, we describe a framework for specifying communicative multi-agent systems, using a theory of action based in the situation calculus to describe the effects of actions on the world and on the mental states of agents; and the concurrent, logic programming language ConGolog to specify the actions performed by each agent. Since ConGolog has a well-defined semantics in the situation calculus, the specifications can be used to reason about the behavior of individual agents and the system as a whole. We extend the work presented in (Lespérance *et al.* 1996) to allow the specifications to mention agents' goals explicitly. The framework presented here allows the behavior of different agents to be specified at different levels of abstraction, using a rich set of programming language constructs. As an example, we specify a meeting scheduler multi-agent system.

1 Introduction

Many agent theories (some examples are (Cohen & Levesque 1990a; 1990b; Rao & Georgeff 1991; Singh 1994)) follow a similar methodology. They present a framework for representing the mental states of agents and the physical states of the world and axiomatize the relationships between the various components (beliefs, goals, intentions, etc.) of the agents' mental states. They then postulate that if the agent has a certain form of commitment to its intentions or goals, it is able to achieve them, and it is adhering to some principles of rationality, then it will act in service of these intentions and eventually achieve them. While this methodology can lead to some interesting theories, the facts that one can conclude from them are quite weak. One may be able to surmise that the agent will eventually achieve its intentions, but not when or by what means. It is not at all clear that these theories can be used to represent and reason about the actual behavior of agents in a detailed way, especially in the context of *complex* multi-agent systems with communicating agents.

On the other hand, there are formalisms for representing and reasoning about communicating, concurrent processes (Fisher 1995; Hoare 1985). However, the representations of these formalisms are typically at a low level. They do not allow the specification of agents in terms of their mental states, nor do they explicitly represent actions in terms of their effects on the world.

In this paper, we explore a middle ground. We use the situation calculus (McCarthy & Hayes 1979) with Reiter's solution to the frame problem (Reiter 1991)—enhanced with predicates to describe agents' knowledge (Scherl & Levesque 1993) and goals—to formally, perspicuously, and systematically describe the effects of actions on the world and the mental states of agents. We add INFORM and REQUEST actions as primitive situation calculus actions to model inter-agent communication. We then use the notation of the concurrent, logic programming language ConGolog (De Giacomo, Lespérance, & Levesque 1997) to specify the behavior of agents. Since ConGolog has a well-defined semantics in the situation calculus, the specifications can be used to reason about the behavior of individual agents and the system as a whole. Since ConGolog is based in the situation calculus, it is easy to add the capacity to specify the behavior of agents in terms of their mental states. In addition, ConGolog offers a wide variety of programming-language constructs (e.g., nondeterministic choice of actions and iteration, waiting for arbitrary first-order formulae to hold, concurrent execution with different priorities, and interrupts), which make it easy to specify a wide range of complex behaviors. One advantage of this approach is that different agents can be specified at different levels of abstraction. In the example system presented in section 6, we model the behavior of one agent only in terms of its knowledge and another in terms of its knowledge and goals.

This paper extends the work presented in (Lespérance *et al.* 1996). In that paper, the agents' mental states consisted only of knowledge. We have added the capacity to specify the behavior of agents in terms of their goals. This addition allows us to dispense with the explicit representation of message queues for agents introduced in (Lespérance *et al.* 1996), yielding a cleaner account of communicative actions. We reformulate the meeting scheduler application described in (Lespérance *et al.* 1996) in this enhanced framework.

2 Theory of action

Our action theory is based on an extended version of the situation calculus (Reiter 1991), a predicate calculus dialect for representing dynamically changing worlds. In this formalism, the world is taken to be in a certain situation. That situation can only change as a result of an agent performing an action. The term $do(a, s)$ represents the situation that results from the agent's executing action a in situation s. For example, the formula $\textsc{On}(\mathrm{A}, \mathrm{B}, do(\textsc{putOn}(\mathrm{A}, \mathrm{B}), s))$ could mean that A is on B in the situation resulting from the agent's doing $\textsc{putOn}(\mathrm{A}, \mathrm{B})$ in s. Predicates and function symbols whose value may change from situation to situation (and whose last argument is a situation) are called *fluents*. There is also a special constant, S_0, used to denote the initial situation.

An action is specified by first stating the conditions under which it can be performed by means of a *precondition axiom*. For example,[1]

$$\begin{aligned}&Poss(\textsc{pickUp}(x), s) \Leftrightarrow\\ &\quad \forall z(\neg\textsc{Holding}(z, s) \wedge \textsc{nextTo}(x, s))\end{aligned}$$

means that it is possible for the agent to pick up an object x in situation s iff it is not holding anything and it is standing next to x in s.

We adopt Reiter's (Reiter 1991) solution to the frame problem,[2] which shows how one can transform axioms defining the effects of actions on fluents into a set of *successor state axioms*, one for each fluent, that imply the effect axioms as well as all the frame axioms. Successor state axioms have the following form, for each fluent, R:

$$\begin{aligned}&[Poss(a, s) \Rightarrow\\ &\quad [R(\vec{x}, do(a, s)) \Leftrightarrow\\ &\qquad \gamma_R^+(\vec{x}, a, s) \vee (R(\vec{x}, s) \wedge \neg\gamma_R^-(\vec{x}, a, s))]],\end{aligned}$$

where $\gamma_R^+(\vec{x}, a, s)$ ($\gamma_R^-(\vec{x}, a, s)$, respectively) denotes a formula which defines the conditions under which R will be true (false, respectively) after performing a in s.

3 Time

For our example application, we will need to model time: what has happened, what will happen, the actual time at a situation, and periods of time. In the situation calculus, the situations are structured in a tree (or a forest when there are several possible initial situations, as is the case here). Each situation has a linear past and a branching future. The past of a situation is well defined. It is simply the (unique) sequence of situations from the initial situation (the root of the tree) to the current situation. **Previously**(a, s) is true if the action a occurred in the past of s:[3]

$$\textbf{Previously}(a, s) \stackrel{\text{def}}{=} \exists s'(do(a, s') \leq s)$$

To model the future of a situation, we need a way to select one future among the many possible futures. We do this using *action selection functions* (ASFs) which are simply functions from situations to actions. Given a starting situation, an ASF defines an infinite path of situations. For each situation in the path, the ASF selects the action that is to be performed next to yield the next situation in the path. We define the predicate **OnPath** to mean that situation s' is in the situation sequence defined by ASF σ and situation s:

$$\begin{aligned}&\textbf{OnPath}(\sigma, s, s') \stackrel{\text{def}}{=}\\ &\quad s \leq s' \wedge \forall a, s^*(s < do(a, s^*) \leq s' \Rightarrow \sigma(s^*) = a).\end{aligned}$$

We can use **OnPath** to define predicates that talk about the future. For example, we could define **Eventually**(ψ, σ, s) to mean that eventually ψ will hold along the path defined by σ starting at s:[4]

$$\begin{aligned}&\textbf{Eventually}(\psi, \sigma, s) \stackrel{\text{def}}{=}\\ &\quad \exists s^*(\textbf{OnPath}(\sigma, s, s^*) \wedge \psi(\sigma, s^*)).\end{aligned}$$

We also introduce a functional fluent $\textsc{time}(s)$ which maps a situation into the current time at a situation. We assume that all actions in the domain increment $\textsc{time}$ by $\textsc{duration}(a)$, the amount of time it takes to perform the action a.

We also model periods of time, since meetings take place over periods of time. A *period* is a pair of times. The first and second elements of the pair are denoted by $\textsc{start}(\textsc{period})$ and $\textsc{end}(\textsc{period})$, respectively. We use the predicate **During**$(period, \psi, \sigma, s)$ to stipulate that ψ holds throughout *period* on the path defined by σ and s:

$$\begin{aligned}&\textbf{During}(period, \psi, \sigma, s) \stackrel{\text{def}}{=}\\ &\quad \exists s', s''(\textbf{OnPath}(\sigma, s, s') \wedge \textbf{OnPath}(\sigma, s, s'') \wedge\\ &\quad \textsc{time}(s') \leq \textsc{start}(period) < \textsc{time}(do(\sigma(s'), s')) \wedge\\ &\quad \textsc{time}(s'') < \textsc{end}(period) \leq \textsc{time}(do(\sigma(s''), s'')) \wedge\\ &\quad \forall s^*(s' \leq s^* \leq do(\sigma(s''), s'') \Rightarrow \psi(\sigma, s^*))).\end{aligned}$$

[1]We adopt the convention that unbound variables in a formula are universally quantified in the widest scope.

[2]For treatments of the ramification and qualification problems compatible with our approach see (Lin & Reiter 1994; McIlraith 1997).

[3]$s < s'$ means that there is a possible (as defined by the *Poss* predicate extended to action sequences) sequence of actions that can be performed starting in situation s and which results in situation s'. $s \leq s'$ is an abbreviation for $s < s' \vee s = s'$.

[4]We use ψ to denote a formula whose fluents contain two placeholders sit and asf instead of a situation argument and an ASF argument (respectively), e.g., $\textsc{Next}(a, \mathsf{asf}, \mathsf{sit})$. The placeholders get replaced by a situation and an ASF by an outer construct, in this case **Eventually**. $\psi(\sigma, s)$ is the formula that results from replacing sit with s and asf with σ. Where the intended meaning is clear, we suppress the placeholders, e.g., $\textsc{Next}(a)$.

Since actions are of extended duration, the endpoints of a time period might fall between two adjacent situations on the path. Therefore, the definition stipulates that ψ holds starting at the latest situation (s') whose time is earlier than or equal to the start of the period until the earliest situation ($do(\sigma(s''), s'')$ whose time is later than or equal to the end of period. Note that in this definition and elsewhere, we overload the $<$ and $\leq$ operators, using them to relate times as well as situations. For other approaches to adding time to the situation calculus, see (Pinto & Reiter 1993; Reiter 1996).

4 Mental Attitudes

4.1 Knowledge

Moore (Moore 1985) showed how agents' knowledge could be modelled in the situation calculus by adapting the possible worlds model of knowledge to the situation calculus. We adopt his approach, using the notation of (Scherl & Levesque 1993). $K(agt, s', s)$ will be used to denote that in situation s, *agt* thinks that it could be in situation s'. We call s' a *K-alternative situation* for *agt* in s. $\mathbf{Know}(agt, \phi, s)$[5] is then defined to be $\forall s'(K(agt, s', s) \Rightarrow \phi(s'))$, and $\mathbf{KWhether}(agt, \phi, s)$ is defined as $\mathbf{Know}(agt, \phi, s) \vee \mathbf{Know}(agt, \neg\phi, s)$.

Scherl and Levesque (Scherl & Levesque 1993) show how to obtain a successor state axiom for K that completely specifies how knowledge is affected by actions. In their framework, the knowledge-producing actions were performed by the agent itself, i.e., sensing actions. In this paper, we model communicative actions, which are actions that affect the mental state (knowledge, goals, commitments, etc.) of an agent other than the one performing the action. However, Scherl and Levesque's successor state axiom for K is easily adapted to handle communicative actions:[6]

$$\begin{array}{l}
Poss(a, s) \Rightarrow \\
[K(agt, s'', do(a, s)) \Leftrightarrow \\
\quad \exists s'(K(agt, s', s) \wedge s'' = do(a, s') \wedge Poss(a, s') \wedge \\
\qquad \forall informer, \phi(a = \textsc{inform}(informer, agt, \phi) \wedge \\
\qquad\qquad\qquad \textsc{Trusts}(agt, informer, \phi) \wedge \\
\qquad\qquad\qquad \neg\mathbf{Know}(agt, \neg\phi, s) \Rightarrow \phi(s')))]
\end{array}$$

First note that for any action other than an INFORM action directed towards *agt*, the specification ensures that the only change in knowledge that occurs in moving from s to $do(a, s)$ is that it is known (by all agents) that the action a has been successfully performed. This is true because the K-alternative worlds to $do(a, s)$ are the situations that result from doing a in a K-alternative situation to s where a is possible. Note that unlike (Lespérance *et al.* 1996), we assume that all actions are public, i.e., every agent is aware of the actions performed by all other agents. For the action $\textsc{inform}(informer, agt, \phi)$,[7] the idea is that in moving from s to $do(\textsc{inform}(informer, agt, \phi), s)$, *agt* not only knows that the action has been performed (as above), but also if *agt* trusts *informer* with respect to ϕ and *agt* does not know $\neg\phi$ in s, then it knows ϕ. In this case, the K-alternative worlds to $do(a, s)$ for *agt* are the situations that result from performing a in a K-alternative world to s where a is possible, except the ones where ϕ is false. If *agt* knows $\neg\phi$ in s or it does not trust *informer*, then the result is that *agt*'s knowledge is unchanged except for being aware that the action has occurred.

TRUSTS would be axiomatized by the modeller of the system to capture the idea that an agent will not necessarily believe everything that another agent tells it. For example, if an agent wants to cancel a meeting, one way to do that would be to inform the organizer of the meeting that another agent cannot attend the meeting, even if that is not true. Thus, we might want to say, for example, that the organizer will only believe an agent if it is reporting something about its *own* mental state. This is a fairly simple-minded account of communicative interaction; we plan to refine the account in the future.

4.2 Goals

We extend the framework described in (Lespérance *et al.* 1996) by modelling the goals of agents. The behavior of agents will be specified, in part, in terms of their own goals and what they know about the goals of other agents. Following (Cohen & Levesque 1990a), we characterize the goals of an agent by specifying the paths in which all its goals (both maintenance goals and achievement goals) are achieved. Our approach differs slightly from Cohen and Levesque's in that our accessibility relation, $H(agt, \sigma, s', s)$, is used to state that in situation s, the path defined by σ and s' is consistent with what *agt* wants. We think of these paths as defining what the agent wants independently of what the agent knows. Cohen and Levesque restricted the G-accessible worlds (their version of H) to be a subset of the B-accessible worlds (their version of K). The end result is the same, however, as we define the goals of the agent to be the formulae true in the H-accessible paths that begin in a K-accessible situation:

$$\begin{array}{l}
\mathbf{Goal}(agt, \psi, s) \overset{\text{def}}{=} \\
\quad \forall \sigma, s'(K(agt, s', s) \wedge H(agt, \sigma, s', s) \Rightarrow \psi(\sigma, s'))
\end{array}$$

[5] We use ϕ to denote a formula that may contain the placeholder sit, but not asf. Again, we suppress the placeholder where possible. $\phi(s)$ is the formula that results from substituting s for sit (if present) in ϕ.

[6] In our example system, we have no need for sensing actions, but they could easily be added if necessary. Also note that here, for simplicity, TRUSTS and SERVES (defined below) are not fluents, and that the axioms that define them are not given.

[7] Since our language contains functions (e.g., INFORM) and predicates that take a formula as an argument, we assume that we have an axiomatization of formulae as terms of the language, such as the one given in (Moore 1985).

We provide a successor state axiom for H similar to the one for K:

$$\begin{array}{l} Poss(a,s) \Rightarrow \\ {[H(agt,\sigma,s'',do(a,s)) \Leftrightarrow} \\ \quad \exists s'(H(agt,\sigma,s',s) \wedge s''=do(\sigma(s'),s') \wedge \\ \qquad \forall requester, \psi(a = \text{REQUEST}(requester, agt, \psi) \wedge \\ \qquad\qquad \text{SERVES}(agt, requester, \psi) \wedge \\ \qquad\qquad \neg\mathbf{Goal}(agt,\neg\psi,s) \Rightarrow \psi(\sigma,s'))] \end{array}$$

Thus, the goals[8] of an agent are changed by REQUEST actions. The SERVES predicate is analogous to the TRUSTS predicate and is used to characterize which requesters and requests the agent will serve. As with knowledge, goals are only added if they are consistent with the agent's previous goals, and once an agent adopts a goal, it maintains the goal indefinitely. Therefore, the agents have a fanatical commitment to their goals. Even the agent that made the request in the first place cannot cause the requested agent to drop the goal that resulted from the request. In future work, we plan to devise a method for goal revision in order to be able to relax this constraint.

4.3 Constraints

Scherl and Levesque (Scherl & Levesque 1993) showed that one could place constraints on the K-relation in the initial situation[9], and the constraints would continue to hold after any (possible) sequence of actions because the successor state axiom for K preserves these constraints. Since we are modelling knowledge (i.e., true beliefs), we constrain K to be reflexive, and we also want it to be transitive and symmetric so the agents will have positive and negative introspection. The only constraint on H in isolation is that it be initially non-empty, i.e., the agent's wishes are initially consistent.

We also need to consider the constraints that hold between H and K. In our example, we want the agents to have positive and negative introspection of goals, i.e.:

$$\begin{array}{l} \mathbf{Goal}(agt,\psi,s) \Rightarrow \mathbf{Know}(agt,\mathbf{Goal}(agt,\psi),s) \\ \neg\mathbf{Goal}(agt,\psi,s) \Rightarrow \mathbf{Know}(agt,\neg\mathbf{Goal}(agt,\psi),s) \end{array}$$

We have identified a constraint on K and H in the initial situation that yields these properties. See the full paper for details.

5 ConGolog

We have just presented a framework in which one can systematically and perspicuously describe the effects of actions on the world and on the mental states of multiple, communicating agents. In order to describe a multi-agent system, we must also specify what actions the agents perform.

We specify the behavior of agents with the notation of the logic programming language ConGolog (De Giacomo, Lespérance, & Levesque 1997), the concurrent version of Golog (Levesque *et al.* 1997). While versions of both Golog and ConGolog have been implemented, we are mainly interested here in the potential for using ConGolog as a specification language. A ConGolog program[10] is composed of a sequence of procedure declarations, followed by a complex action. The complex actions are composed of various constructs:

a,	primitive action
$\phi?$,	wait for a condition
$(\delta_1;\delta_2)$,	sequence
$(\delta_1 \mid \delta_2)$,	nondeterministic choice between actions
δ^*,	nondeterministic iteration
if ϕ **then** δ_1 **else** δ_2,	conditional
for $x \in \Sigma$ **do** δ,	for loop
while ϕ **do** σ,	while loop
$(\delta_1 \parallel \delta_2)$,	concurrent execution
$(\delta_1 \rangle\!\rangle \delta_2)$,	concurrency with different priorities
$\langle\ \vec{x} : \phi \rightarrow \delta\ \rangle$,	interrupt
$\beta(\vec{p})$,	procedure call.

a denotes a situation calculus action, as described above. The ConGolog specification can be for a single agent or multiple agents, depending on whether the primitive actions contain an argument for the agent of the action. ϕ denotes a situation calculus formula with the situation argument of its fluents suppressed. δ, δ_1, and δ_2 stand for complex actions, Σ is a set, β is a procedure name, and $\vec{p}$ are the actual parameters to the procedure.

These constructs are mostly self-explanatory. Intuitively, the interrupts work as follows. Whenever $\exists\vec{x}.\phi$ becomes true then δ is executed with the bindings of $\vec{x}$ that satisfied ϕ.

Procedures are defined with the following syntax: **proc** $\beta(\vec{x})\ \delta$, where β is the procedure name, $\vec{x}$ are the formal parameters to the procedure, and δ is a complex action. ConGolog programs are formally defined using the Do predicate (see (De Giacomo, Lespérance, & Levesque 1997) for details). Informally, $Do(\rho,s,s')$ holds if situation s' is a legal terminating situation of program ρ, starting in situation s.

6 Specification of a Meeting Scheduler Multi-Agent System

In our example system, we will have meeting organizer agents, which are trying to schedule meetings, and personal agents, which manage the schedules of their (human) owners. To simplify the system, we assume that the personal agents have the authority to schedule meetings on behalf of their owners without

[8]We use the term 'goal' loosely here to refer to the formulae true in all H-related worlds.

[9]In particular, they show that K can be constrained to be reflexive, symmetric, transitive, and/or Euclidean.

[10]We retain the term 'program' even though it is not our intention to execute the programs directly.

consulting them. They simply inform their owners of a pending meeting fifteen minutes before it starts. The meeting organizer agents have the built-in behavior to schedule a single meeting and then inform the (human) chair of the meeting whether the meeting was successfully scheduled. A meeting is successfully scheduled if all the personal agents agree to have their owners attend the meeting, otherwise the meeting is cancelled.

A significant feature of our approach is that different agents can be modelled at different levels of generality. The personal agents are specified in terms of their knowledge and their goals, whereas the behavior of the meeting organizer agents depends only on their knowledge. In other contexts, we might find it useful to model some agents without referring to their mental state at all.

In order to specify our multi-agent system, we must first define the actions and fluents of our domain. The actions are INFORM and REQUEST which were previously described, and

- GOTOMEETING($user, chair$): (the human) $user$ goes to a meeting chaired by (the human) $chair$,
- LEAVEMEETING($user, chair$): $user$ leaves the meeting chaired by $chair$, and
- TICK: the time increases by one minute.

The fluents are K, H, and TIME, which were previously described, and ATMEETING($user, chair, s$), which means that $user$ is at a meeting chaired by $chair$ in s. In addition, we have two non-fluent functions: PAG($user$), whose value is the personal agent of $user$, and DURATION(a), which maps an action to its duration.

Here are the axioms defining the actions and fluents (other than the successor state axioms for K and H which were stated previously):

Precondition axioms:

$$\begin{aligned}
&Poss(\textsc{inform}(informer, agt, \phi), s) \Leftrightarrow \\
&\quad \mathbf{Know}(informer, \phi, s) \\
&Poss(\textsc{request}(requester, agt, \psi), s) \Leftrightarrow \\
&\quad \mathbf{Goal}(requester, \psi, s) \\
&Poss(\textsc{gotoMeeting}(user, chair), s) \Leftrightarrow \\
&\quad \neg\exists chair'(\textsc{AtMeeting}(user, chair', s)) \\
&Poss(\textsc{leaveMeeting}(user, chair), s) \Leftrightarrow \\
&\quad \textsc{AtMeeting}(user, chair, s) \\
&Poss(\textsc{tick}, s)
\end{aligned}$$

Successor state axioms:[11]

$$\begin{aligned}
&Poss(a, s) \Rightarrow \\
&\quad [\textsc{AtMeeting}(user, chair, do(a, s)) \Leftrightarrow \\
&\qquad a = \textsc{gotoMeeting}(user, chair) \vee \\
&\qquad\quad (\textsc{AtMeeting}(user, chair, s) \wedge \\
&\qquad\quad\; a \neq \textsc{leaveMeeting}(user, chair))] \\
&Poss(a, s) \Rightarrow \\
&\quad [\textsc{Time}(do(a, s)) = t \Leftrightarrow \\
&\qquad \textsc{Time}(s) = t' \wedge t = t' + \textsc{duration}(a)]
\end{aligned}$$

[11]For simplicity, we omit the axioms that define DURATION(a) for each action a.

Initial state axioms:

$$\begin{aligned}
&\mathbf{Know}(agt, \textsc{time} = 9{:}00 \wedge \\
&\quad \neg\textsc{AtMeeting}(user, chair), S_0)
\end{aligned}$$

We are now ready to use ConGolog to define the behavior of the agents. We start with the meeting organizer agents. Their task is to organize a meeting for a given period and set of participants on behalf of the chair of the meeting. They do this by asking the personal agent of each participant to be prepared to have their owner meet (see below for a discussion) during the given period. The participants' agents will then reply whether they have adopted the goal to have their owners meet at the appointed time. In our example, the personal agents adopt the goal to meet iff they do not have a scheduling conflict. The meeting organizer waits until it knows whether one of the participants' agent declined the meeting. This will happen when either someone has declined the meeting or everyone has agreed to it. Once the organizer knows the answer, it informs all the participants' agents and the chair whether someone has declined.

When a meeting organizer requests a meeting from a personal agent, we would like the request to be simply that the personal agent's owner attend the meeting at the appropriate time. However, we were not able to model the system this way, since we are lacking a method to handle goal revision. If the organizer agent were to simply request a meeting, then any personal agent involved in the meeting that did not have prior plans for its owner during the period of the meeting would adopt the goal that its owner go to the meeting. If, later, the meeting is cancelled, there would be no way to get the agent to drop the goal, so the agent would continue to refuse requests for other meetings at that time.

To get around the need for goal revision, the organizing agent requests the participants' personal agents to adopt a disjunctive goal (which is formalized in figure 1 as PREPAREDTOMEET): that the participant attend the meeting, or that *some* participant's personal agent declines the meeting. Suppose a personal agent adopts this goal. If no one declines to meet, then the organizing agent informs the personal agent that no one declined to meet. Once the personal agent knows this, then the second disjunct of the goal is falsified in the personal agent's K-accessible worlds, therefore the agent ends up having the goal that its owner attend the meeting. This implies that the personal agent has the goal that its owner does not attend any other meeting that happens at the same time. Therefore, the personal agent will decline any such meeting. If the organizing agent informs the personal agent that someone declined, then the agent does not end up with the goal that its owner attend the meeting.

The procedure that defines the behavior of the organizer agent is given in figure 1. We assume that definition of PREVIOUSLY from section 3 has been expanded to take complex actions as arguments as well as primitive actions. The arguments to the procedure are the organizing agent, the chair of the meeting, the set of participants with whom the chair wants to meet, and the time period of the meeting.

The procedure that specifies the behavior of the personal agents is given in figure 2. Its arguments are the personal agent and its owner. The procedure is defined with two interrupts, the first running at higher priority than the second. The first interrupt fires when the agent has the goal that its owner be at a meeting that starts in less than fifteen minutes, and the agent knows that it has not previously informed the user to go to the meeting. The action taken by the agent is to inform the agent that it "wants" the user to go the meeting.

The second interrupt handles meeting requests; it fires when the agent has the goal to be prepared to have its owner attend a meeting, and it knows that it has not previously informed the organizer of the meeting that it has this goal. The action taken is to inform the organizer whether it has the goal that its owner not attend the meeting, i.e., whether it already has the goal that its owner attend another meeting at the same time. This interrupt works as required because if the agent has committed to a meeting and a request arrives for another meeting at the same time, the agent still adopts the disjunctive goal PREPAREDTOMEET(...). The first disjunct of the new goal conflicts with the agent's previous goals, but the second disjunct does not because the agent will not yet know whether someone declined the meeting. In this case, the agent will inform the organizer that it declines the meeting.

A complete meeting scheduler system is modelled by composing instances of the two agent procedures in parallel, thereby modelling the behavior of several agents acting independently.[12] We also need to compose the nondeterministic iteration of the tick action in parallel with the calls to the agent procedures to allow time to pass when the agents are not acting. For example, let the program ρ consist of the three procedure definitions given in figures 1 and 2, followed by:

```
[MANAGESCHEDULE(PA_1, USER_1) ||
MANAGESCHEDULE(PA_2, USER_2) ||
MANAGESCHEDULE(PA_3, USER_3) ||
SCHEDULEMEETING(OA_1, USER_1, {USER_1, USER_3},
                12:00–2:00) ||
SCHEDULEMEETING(OA_2, USER_2, {USER_2, USER_3},
                1:30–2:45)]
⟫ TICK*
```

A sequence of actions $\vec{a}$ that satisfies

$$Axioms \models Do(\rho, S_0, do(\vec{a}, S_0))$$

[12]The model is only approximate, since the *Do* relation defines interleaved executions of parallel actions, whereas one would expect that in practice different agents will be running on different processors allowing them to act simultaneously.

will represent a possible evolution of the system.

In this example, the meeting schedulers will both try to obtain USER$_3$'s agreement for meetings that overlap; there will thus be two types of execution sequences, depending on who obtains this agreement. In the full paper, we give examples of sequences of actions that satisfy this specification, and we prove properties of the agents whose behavior satisfies the specification.

7 Discussion and Future Work

We have stressed that we are using ConGolog programs as specifications of agent behavior that can be used to prove properties of multi-agent systems rather than as implementations of agents. However, since ConGolog is a programming language, we could simulate the behavior of a system by running its program. A ConGolog interpreter has been implemented. Unfortunately, the interpreter does not handle queries about goals or knowledge. However, if the interpreter had access to a first-order logic theorem prover (since our definitions of **Know** and **Goal** are first-order), the program defined in section 6 could be run (though not very efficiently).

If we did run the program, we would have a simulation of the multi-agent system. What we really want, however, is a method of transforming the program into a set of programs, one for each agent, that could be executed on different processors to yield a true multi-agent system that implements the original specification. Since the program is composed of separate procedures for each agent, this should not be too hard to accomplish. However, once again, we would have to find a way of dealing with queries about the mental states of agents, or try to find a transformation that could produce single-agent programs that do not refer to mental states. It would be interesting to identify a large class of multi-agent ConGolog programs that could be decomposed into single-agent ConGolog programs, and show that properties that were proven about the multi-agent program ought still to hold for a system composed of the agents running the single-agent programs on separate processors.

The multi-agent meeting scheduler application described above is very simplistic. We would like to expand it by modelling more complex communicative interactions between the agents, and between the agents and their owners. In order to accomplish this, we would have to develop a more sophisticated theory of communicative interaction than the one presented in section 4. Also, we want to develop a method for handling goal (and belief) revision in our framework.

References

Cohen, P. R., and Levesque, H. J. 1990a. Intention is choice with commitment. *Artificial Intelligence* 42:213–261.

Cohen, P. R., and Levesque, H. J. 1990b. Rational interaction as the basis for communication. In Cohen,

```
proc $\text{ORGANIZEMEETING}(oa, chair, Participants, period)$
  for $p \in Participants$ do
    $\text{REQUEST}(oa, \text{PAG}(p), \text{PREPAREDTOMEET}(p, period, chair, Participants))$;
  $\mathbf{KWhether}(oa, \text{SOMEONEDECLINED}(period, chair, Participants))?$;
  for $p \in Participants$ do
    $\text{INFORMWHETHER}(oa, \text{PAG}(p), \text{SOMEONEDECLINED}(period, chair, Participants))$;
  $\text{INFORMWHETHER}(oa, chair, \text{SOMEONEDECLINED}(period, chair, Participants))$;
endProc,
```

where:

$$\text{SOMEONEDECLINED}(period, chair, Participants) \overset{\text{def}}{=}$$
$$\exists p \in Participants(\mathbf{Goal}(\text{PAG}(p), \mathbf{During}(period, \neg \text{ATMEETING}(p, chair))))$$
$$\text{PREPAREDTOMEET}(p, period, chair, Participants) \overset{\text{def}}{=}$$
$$\mathbf{During}(period, \text{ATMEETING}(p, chair)) \lor \text{SOMEONEDECLINED}(period, chair, Participants).$$

```
proc $\text{INFORMWHETHER}(agt, agt', \phi)$
  $\mathbf{Know}(agt, \phi)?; \text{INFORM}(agt, agt', \phi)$ |
  $\mathbf{Know}(agt, \neg\phi)?; \text{INFORM}(agt, agt', \neg\phi)$ |
  $\neg\mathbf{KWhether}(agt, \phi)?; \text{INFORM}(agt, agt', \neg\mathbf{KWhether}(agt, \phi))$
endProc
```

Figure 1: Procedure run by the meeting organizer agents.

P. R.; Morgan, J.; and Pollack, M. E., eds., *Intentions in Communication.* Cambridge, MA: MIT Press. 221–255.

De Giacomo, G.; Lespérance, Y.; and Levesque, H. J. 1997. Reasoning about concurrent execution, prioritized interrupts, and exogenous actions in the situation calculus. To appear in Proceedings of IJCAI-97.

Fisher, M. 1995. Towards a semantics for Concurrent METATEM. In Fisher, M., and Owens, R., eds., *Executable Modal and Temporal Logics (LNAI Volume 896).* Springer-Verlag.

Hoare, C. 1985. *Communicating Sequential Processes.* Prentice Hall Int.

Lespérance, Y.; Levesque, H. J.; Lin, F.; Marcu, D.; Reiter, R.; and Scherl, R. B. 1996. Foundations of a logical approach to agent programming. In Wooldridge, M.; Müller, J. P.; and Tambe, M., eds., *Intelligent Agents Volume II — Proceedings of the 1995 Workshop on Agent Theories, Architectures, and Languages (ATAL-95)*, Lecture Notes in Artificial Intelligence. Springer-Verlag. 331–346.

Levesque, H. J.; Reiter, R.; Lespérance, Y.; Lin, F.; and Scherl, R. B. 1997. GOLOG: A logic programming language for dynamic domains. *Journal of Logic Programming* 31:59–84.

Lin, F., and Reiter, R. 1994. State constraints revisited. *Journal of Logic and Computation* 4(5):655–678.

McCarthy, J., and Hayes, P. 1979. Some philosophical problems from the standpoint of artificial intelligence. In Meltzer, B., and Michie, D., eds., *Machine Intelligence*, volume 4. Edinburgh, UK: Edinburgh University Press. 463–502.

McIlraith, S. A. 1997. *Towards a Formal Account of Diagnostic Problem Solving.* Ph.D. Dissertation, Department of Computer Science, University of Toronto, Toronto, ON.

Moore, R. C. 1985. A formal theory of knowledge and action. In Hobbs, J. R., and Moore, R. C., eds., *Formal Theories of the Common Sense World.* Norwood, NJ: Ablex Publishing. 319–358.

Pinto, J., and Reiter, R. 1993. Adding a time line to the situation calculus. In *The Second Symposium on Logical Formalizations of Commonsense Reasoning*, 172–177.

Rao, A. S., and Georgeff, M. P. 1991. Modeling rational agents within a BDI-architecture. In Fikes, R., and Sandewall, E., eds., *Proceedings of Knowledge Representation and Reasoning (KR&R-91)*, 473–484. Morgan Kaufmann Publishers: San Mateo, CA.

Reiter, R. 1991. The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result for goal regression. In Lifschitz, V., ed., *Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy.* San Diego, CA: Academic Press. 359–380.

Reiter, R. 1996. Natural actions, concurrency and continuous time in the situation calculus. In *Proc. of the 5th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'96)*, 2–13.

```
$\mathbf{proc}$ $\textsc{manageSchedule}(pa, user)$
  $\langle period, chair$ :
    $\mathbf{Goal}(pa, \mathbf{During}(period, \textsc{AtMeeting}(user, chair))) \wedge$
    $\mathbf{Know}[pa, \neg\mathbf{Previously}(\textsc{inform}(pa, user, \mathbf{Goal}(pa, \mathbf{During}(period, \textsc{AtMeeting}(user, chair))))) \wedge$
              $\textsc{start}(period) - :15 \leq \textsc{time} \leq \textsc{start}(period)] \rightarrow$
                $\textsc{inform}(pa, user, \mathbf{Goal}(pa, \mathbf{During}(period, \textsc{AtMeeting}(user, chair)))) \rangle$
  $\rangle\!\rangle$
  $\langle oa, period, Participants, chair$ :
    $\mathbf{Goal}(pa, \textsc{PreparedToMeet}(user, period, chair, Participants)) \wedge$
    $\mathbf{Know}[pa, \mathbf{Previously}(\textsc{request}(oa, pa, \textsc{PreparedToMeet}(user, period, chair, Participants))) \wedge$
              $\neg\mathbf{Previously}(\textsc{informWhether}(pa, oa,$
                  $\mathbf{Goal}(pa, \mathbf{During}(period, \neg\textsc{AtMeeting}(user, chair)))))] \rightarrow$
                    $\textsc{informWhether}(pa, oa, \mathbf{Goal}(pa, \mathbf{During}(period, \neg\textsc{AtMeeting}(user, chair)))) \rangle$
$\mathbf{endProc}$
```

Figure 2: Procedure run by the personal agents.

Scherl, R. B., and Levesque, H. J. 1993. The frame problem and knowledge-producing actions. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, 689–695. Washington, DC: AAAI Press/The MIT Press.

Singh, M. P. 1994. *Multiagent Systems: A Theoretical Framework for Intentions, Know-How, and Communications (LNAI Volume 799)*. Springer-Verlag: Heidelberg, Germany.