A Formal Methodology for Verifying Situated Agents

Phan Minh Dung
Department of Computer Science,
Asian Institute of Technology
PO Box 2754, Bangkok 10501, Thailand
dung@cs.ait.ac.th

Abstract

In this paper, we develop a formal methodology for verifying situated agents. The methodology consists of two elements, a specification language for specifying the agent capabilities to execute its actions in dynamic environments and a repertoire of proof methods by which the correctness of an agent, relative to its capabilities, can be formally verified.

Keywords: Planning and control: situated reasoning, plan execution, reactive control.

Introduction

In recent years there is a shift in AI from the classical paradigm of rational agents to the notion of reactive, situated agents that have an intelligent ongoing interaction with dynamic and uncertain environments (Agre et al 1987, Brooks 1991, Georgeff et al 1987, Kowalski et al 1996, Rosenschein et al 1995, Saffioti et al 1995, Shoham 1993). The correct construction of reliable situated agents is an important task in agent research nowadays.

Consider for example the following plan for a robotic agent to cross a busy motor way:

Wait until the road is clear then cross it.

Is it a correct plan to cross a busy road ? [1]

The correctness of this plan depends very much on the robot's capabilities. If the robot is fast enough to be able to finish crossing the road before a car could pass by then the above plan is correct. But it may not be correct for a slower robot.

The correctness of plans depends on the capabilities of the agents executing them. There is some debate in the literature on whether capability should be defined in mental terms or not (Cohen et al 1990, Lesperance et al 1995, Shoham, 1993, Thomas 1994). Here we will not take a stance on this issue. Our notion of capability is grounded fully in terms of actions and states of automata-like agents. We define an agent's capability as sets of possible execution processes that can unfold during the execution of the agent's plans in interaction with the environment.

Let us look at the road crossing problem again. As crossing a road takes time, we follow the literature (Reiter 1996) in representing it as a sequence of two actions: *start-crossing;end-crossing* where the effects of *start-crossing* and *end-crossing* are *on-road* and *not on-road*, respectively.

For an agent to execute a plan in an uncertain environment is like to play a game with an unpredictable opponent where the moves of the agent are constrained by the plan while the environment can make its moves randomly. In the road crossing game, the environment has two actions *car-appear* and *car-pass-by* to its disposal where the action *car-appear* is executable in all situations (i.e. a car could appear anytime) with the effect that the fluent *car-coming* becomes true after a car appears. The action *car-pass-by* is executable only in those situations in which *car-coming* is true

Imagine a situation s where the agent is alive and on road while a car is coming. Formally s is represented by $s = \{alive, on\text{-}road, car\text{-}coming\}$. Further let p be a plan whose only action is *end-crossing*. Let us consider two possible scenarios in the game to execute this plan between the agent and the environment. In one scenario, the agent manages to make a move by executing the action *end-crossing* before the environment does anything. This scenario is represented by the sequence of state transitions: $s \xrightarrow{end\text{-}crossing} s'$ where $s' = \{alive, car\text{-}coming\}$. In the other scenario, the environment manages to make a move by executing the action *car-pass-by* before the agent could finish the action *end-crossing*. Afterwards the agent's action *end-crossing* is not defined anymore (as the agent has ceased to be alive). This scenario is represented by the sequence of state transitions: $s \xrightarrow{car\text{-}pass\text{-}by} s'' \xrightarrow{\perp} s''$ where $s'' = \{on\text{-}road\}$ and the label $\perp$ means that the game has been interupted as the agent can not execute the action *end-crossing* any more.

Note that though both actions *end-crossing* and *car-pass-by* are executable in the situation s, the second

[1] By correctness we mean that the robot should not be run over by cars when executing this plan

scenario can not happen if the agent has the capability to finish crossing the road before any car could pass by. [2]

Formally, an agent's capability is represented by an capability function $\mathcal{C}$ which assigns to each plan p and state r, a set $\mathcal{C}(p, r)$ of possible execution processes of p, starting from r, which could unfold in the agent's game with the environment. For example, the agent's capability to finish crossing a road before a car could pass by in the situation s above is characterized by $\mathcal{C}(end\text{-}crossing, s) = \{s \overset{end\text{-}crossing}{\longrightarrow} s'\}$

In this paper, we develop a formal methodology for verifying situated agents. The methodology consists of two elements, a specification language for specifying the agent capabilities to execute its actions in dynamic environments and a repertoire of proof methods by which the correctness of an agent, relative to its capabilities, can be formally verified.

Preliminaries

The methodology for representing actions in this paper is adopted from (Gelfond et al 1993). The language for describing the world consists of a set of propositional fluent names FLU and a set of action names ACT. A state of the world is represented by a subset of FLU. The set of all states is denoted by STA. The effects of an action A is determined by a partial transition function $T_A : STA \rightarrow STA$. We say that A is *executable* in s if $T_A(s)$ is defined.

Example 1 *The language for our road crossing problem is given by:* $FLU = \{$alive (al), on-road (on), car-coming $(cc)\}$, and $ACT = \{$car-appear (ca), car-pass-by (cp), start-crossing (sc), end-crossing $(ec)\}$. *The actions have effects according to their commonsense understanding:*

$$T_{sc}(s) = \begin{cases} s \cup \{on\} & if\ s \models al \wedge \neg on \\ undefined & otherwise \end{cases}$$

$$T_{ec}(s) = \begin{cases} s - \{on\} & if\ s \models al \wedge on \\ undefined & otherwise \end{cases}$$

$T_{ca}(s) = s \cup \{cc\}$ *for every s.*

$$T_{cp}(s) = \begin{cases} s - \{cc\} & if\ s \models cc \wedge \neg on \\ s - \{cc, al\} & if\ s \models cc \wedge on \\ undefined & otherwise \end{cases}$$

Reasoning about the effect of actions have been studied extensively in the literature (see Gelfond et all 93, Reiter 96 and the references therein). As our interest in this paper is not primarily about reasoning about action, we will not dwell further on this topic from now on.

[2] We assume that the agents always act with their best capability

Agent Capabilities

The world of our agent in this paper consists of the agent itself and its environment. Therefore the set of action names ACT is a disjoint union $ACT = ACT_{agent} \cup ACT_{env}$ of the set of the agent's actions ACT_{agent} and the set of environment actions ACT_{env}.

A *sequential plan* is defined as a sequence $A_0; \ldots; A_n$ of the agent's atomic actions $A_i \in ACT_{agent}$, $0 \leq i \leq n$.

Definition 1 *1. A (possibly infinite)* global process e *starting from s_0 is a sequence of the form*

$$s_0 \xrightarrow{A_0} s_1 \ldots \xrightarrow{A_n} s_{n+1} \ldots$$

such that for each $i \geq 0$, $A_i \in ACT$ and A_i is executable in s_i, and $s_{i+1} = T_{A_i}(s_i)$. The set of all states in e is denoted by St(e), *i.e.* $St(e) = \{s_0, \ldots, s_n, \ldots\}$.

2. *A global process is called an* environment process *if for each $i \geq 0$, $A_i \in ACT_{env}$*
3. *The initial and final state of a global process e are denoted by initial(e) and final(e) respectively. We often write $s \overset{e}{\leadsto} s'$ to indicate that e is a global process with initial state s and final state s'. We also write $s \overset{e}{\leadsto}$ to indicate that e is a global process with initial state s.*
4. *For global processes e, e' such that final(e) = initial(e'), the concatenation of e and e', denoted by e.e', is defined as the global process*

   $initial(e) \overset{e}{\leadsto} final(e) \overset{e'}{\leadsto}$

   For sets of global processes S,S', define $S.S' = \{e.e' \mid e \in S,\ e' \in S',\ final(e) = initial(e')\}$

To understand the definition of the crucial notion of plan execution process in the following definition, remember that for an agent to execute a plan in an uncertain environment is like to play a game with an unpredictable opponent where the moves of the agent are constrained by the plan while the environment can make its moves randomly.

Definition 2 *A possible execution process of a sequential plan p starting from a state $s_0 \in STA$ is defined inductively as follows:*

1. *$p = A$ where $A \in ACT_{agent}$. Then a possible execution process of p starting from s_0 has*
   - *(a) either the form* $s_0 \overset{e}{\leadsto} s_1 \xrightarrow{A} s_2$ *where e is a finite environment process and A is executable in s_1, and $s_2 = T_A(s_1)$.* [3]
   - *(b) or the form* $s_0 \overset{e}{\leadsto} s_1 \xrightarrow{\perp} s_1$ *where e is a finite environment process and A is not executable in s_1.* [4]

[3] *In this case, the environment has already made the moves in e before the agent manages to actually perform A*

[4] *In this case, the environment has already made the moves in e before the agent is about to perform A which has become unexecutable.*

2. $p = A;q$ where $A \in Act_{agent}$. Then a possible execution process of p starting from s_0 is

(a) either an execution process of A from s_0 which is ended by an undefined action $\perp$

(b) or of the form $e_1.e_2$ where e_1 is a successful execution process of A [5], and e_2 is a execution process of q starting from $final(e_1)$.

3. The set of all possible execution processes of p starting from $s \in STA$ is denoted by Exe(p,s).

4. A plan execution process is interupted if it is ended by an undefined action $\perp$

5. A plan execution process is successful if it is not interupted.

Two examples of possible execution processes of the plan $p = sc; ec$ in the road crossing example are:
$s_0 \xrightarrow{sc} s_1 \xrightarrow{ec} s_0$ and $s_0 \xrightarrow{sc} s_1 \xrightarrow{ca} s_2 \xrightarrow{ec} s_3$
where $s_0 = \{al\}$, $s_1 = \{al, on\}$, $s_2 = \{al, on, cc\}$, $s_3 = \{al, cc\}$

Now we can define the crucial notion of capability.

Definition 3 *The capability of an agent is defined as a function that assigns to each pair (p,s) of a sequential plan p, and a state $s \in STA$ a set $\mathcal{C}(p, s) \subseteq Exe(p, s)$ such that the following properties are satisfied:*

1. $\mathcal{C}(p, s)$ is a finite nonempty subset of $Exe(p, s)$

2. For any plans p_1, p_2: $\mathcal{C}(p_1; p_2) = \mathcal{C}(p_1).\mathcal{C}(p_2)$ where for any plan p, $\mathcal{C}(p) = \bigcup\{\mathcal{C}(p, s) \mid s \in STA\}$.

Intuitively, $\mathcal{C}(p, s)$ represents the set of all possible execution processes which could occur when the agent is executing p from s. The first condition is motivated by the assumption that our agent can execute any executable action $A \in ACT_{agent}$ in a finite time interval, and to exlude *Zeno processes* in which there are infinitely many state-changes in a finite time interval. The motivation for the second condition should be intuitively clear.

In the road crosing example, our agent's capabilities to cross a clear road before any car could appear is represented by any capability function $\mathcal{C}$ satisfying $\mathcal{C}(sc; ec, s_0) = \{s_0 \xrightarrow{sc} s_1 \xrightarrow{ec} s_0\}$. Morerover, the incapability to finish crossing a road alive in the state $s_2 = \{al, on, cc\}$ is expressed by the following capability function: $\mathcal{C}(ec, s_2) = \{s \xrightarrow{cp} r \xrightarrow{\perp} r\}$ where $r = \{on\}$

It is not difficult to see that the following lemma holds.

Lemma 1 *Two capability functions coincide on every plan if they coincide on the single action plans.*

This lemma shows that the capability of an agent to carry out a plan is fully determined by her capability to carry out the basic actions in the plan.

For later use, for each plan p, each fluent proposition φ, define $\mathcal{C}(p, \varphi) = \bigcup\{\mathcal{C}(p, s) \mid s \models \varphi\}$.

[5] *See point 5 in this definition*

A Language for Agent Capabilities

Agent capabilities are described by *ability propositions* of the form

executable p **when** φ **before** ψ

where φ, ψ are fluent propositions and p is a sequential plan. The informal semantics of such proposition is that whenever the agent starts executing p in a state satisfying φ, it will finish before ψ could become true.

For example, the proposition that the agent is capable to cross a clear road before any car could appear, is expressed by the following proposition:

$$\textbf{executable } sc; ec \textbf{ when } \neg cc \wedge al \wedge \neg on \textbf{ before } cc \quad (1)$$

The semantics of the ability propositions is defined with respect to the capability functions as follows.

Definition 4 *Let $\mathcal{C}$ be a capability function and ϵ be an ability proposition of the form **executable** p **when** φ **before** ψ.*

1. We say that $\mathcal{C}$ satisfies *ϵ, written $\mathcal{C} \models \epsilon$, if for each $e \in \mathcal{C}(p, \varphi)$: e is successful and $\forall s' \in St(e)$, $s' \not\models \psi$*

2. $\mathcal{C}$ is said to be a model *of a set of ability propositions if $\mathcal{C}$ satisfies each proposition in this set. A set of ability propositions is* consistent *if it has a model.*

3. We say that ϵ follows *from a set of ability propositions $\mathcal{E}$, written $\mathcal{E} \models \epsilon$ if each model of $\mathcal{E}$ is also a model of ϵ.*

In the road crossing example, any capability function $\mathcal{C}$ such that

$$\mathcal{C}(sc; ec,\, s_0) = \{s_0 \xrightarrow{sc} s_1 \xrightarrow{ec} s_0\} \quad (2)$$

satisfies the ability proposition (1).

Reactive Plans

For each fluent proposition φ, we assume that ACT_{agent} contains an action $\varphi?$ to test whether φ holds in the current situation. Formally, the semantics of a test action $\varphi?$ is defined by

1. $T_{\varphi?}(s)$ is defined iff $s \models \varphi$
2. If $T_{\varphi?}(s)$ is defined then $T_{\varphi?}(s) = s$

Definition 5 *1. A* conditional plan *c has the form*
$\varphi \Longrightarrow p$
where φ, called the test *of c and denoted by test(c), is a fluent proposition and p, called the body of c and denoted by body(c), is a sequential plan.*

2. A reactive plan *is a finite (possibly empty) set of conditional plans. The disjunction of the tests of all the conditional plans in a reactive plan $\mathcal{R}$ is denoted by cond($\mathcal{R}$).*

Reactive plans are executed in cycles. At the begin of each cycle, the test conditions of the conditional plans are checked. If none of them hold then the agent waits until some of them become true. If one of the test conditions holds (either at the first check or after waiting a while), then the body of the corresponding conditional plan is executed. If it is successful then a new cycle starts. If not, the execution of the reactive plan will be interupted.

For an agent to operate safely in an environment, it seems necessary that it should possess a capability to sense any relevant change of the environment early enough to react accordingly. An agent is said to be *alert* if it has the capability to recognize every relevant change of the environment "instantly" (in the sense that during the sensing time the environment undergoes no significant change).

Definition 6 *A capability function $\mathcal{C}$ is said to be* alert *if for any fluent proposition φ*

$$\mathcal{C}(\varphi?, s) = \{s \xrightarrow{\varphi?} s\} \quad \text{if} \quad s \models \varphi$$
$$\mathcal{C}(\varphi?, s) = \{s \xrightarrow{\perp} s\} \quad \text{if} \quad s \models \neg\varphi$$

For simplicity and understandability, we will restrict ourself on alert agents when considering the execution of reactive plans.

Definition 7 *Let $\mathcal{C}$ be an alert capability function and $\mathcal{R} = \{\varphi_0 \Rightarrow p_0, \ldots, \varphi_n \Rightarrow p_n\}$ be a reactive plan.*

1. *A possible execution process e of $\mathcal{R}$ with respect to $\mathcal{C}$ starting from an initial state s_0 is defined as follows:*
   (a) *Case 1: $s_0 \models cond(\mathcal{R})$. Then for some k such that $s_0 \models \varphi_k$,*
      i. *either e has the form $e_0.e_1$ where e_0 is a successful execution process in $\mathcal{C}(\varphi_k?; p_k, s_0)$, and e_1 is an execution process of $\mathcal{R}$ with respect to $\mathcal{C}$ starting from $final(e_0)$.*
      ii. *or e is an interupted execution process in $\mathcal{C}(\varphi_k?; p_k, s_0)$*
   (b) *Case 2: $s_0 \not\models cond(\mathcal{R})$. Then*
      i. *either e has the form $e_0.e_1$ where*
         - *e_0 is an environment process of the form*

$$s_0 \xrightarrow{A_0} s_1 \ldots \xrightarrow{A_{m-1}} s_m$$

           such that for each $0 \le i < m$, $s_i \not\models cond(\mathcal{R})$ and $s_m \models cond(\mathcal{R})$, and
         - *e_1 is an execution process of $\mathcal{R}$ with respect to $\mathcal{C}$ starting from s_m*
      ii. *or e is an infinite environment process such that for each state $s \in St(e)$, $s \not\models cond(\mathcal{R})$ [6].*
      iii. *or e is a finite environment process such that for each $s \in St(e)$ $s \not\models cond(\mathcal{R})$ and there exists no executable environment action at $final(e)$ [7].*
2. *A reactive plan execution process is interupted if it is ended by an undefined action $\perp$.*
3. *A reactive plan execution process is successful if it is not interupted.*
4. *The set of all execution processes of $\mathcal{R}$ wrt $\mathcal{C}$ starting from a state s is denoted by $Exe_{\mathcal{C}}(\mathcal{R}, s)$ while $Exe_{\mathcal{C}}(\mathcal{R}, \varphi)$ denotes the set of all execution processes of $\mathcal{R}$ wrt $\mathcal{C}$ starting from a state satisfying φ*

[6] *This represents the case where the agent has to wait infinitely*

[7] *The agent also has to wait infinitely in this case*

For example, consider again the road crossing example where the agent has the capability (2). Let $\mathcal{R}_0 = \{\neg cc \Rightarrow sc; ec\}$. Then $Exec_C(\mathcal{R}_0, s_0) = \{s_0 \xrightarrow{\neg cc?} s_0 \xrightarrow{sc} s_1 \xrightarrow{ec} s_0 \xrightarrow{\neg cc?} s_0 \xrightarrow{sc} s_1 \xrightarrow{ec} s_0 \longrightarrow \ldots\}$

Verification of Reactive Agents

We consider in this paper the verification of invariance formulas of the forms

$$\varphi \rightarrow \Box(\mathcal{R}, \psi)$$

which states that if the agent starts executing $\mathcal{R}$ (a reactive plan) in a state satisfying φ then ψ will always hold during the execution of $\mathcal{R}$.

We say that an alert capability function $\mathcal{C}$ *satisfies* a formula $\varphi \rightarrow \Box(\mathcal{R}, \psi)$, written $\mathcal{C} \models \varphi \rightarrow \Box(\mathcal{R}, \psi)$, if for each execution process $e \in Exe_{\mathcal{C}}(\mathcal{R}, \varphi)$, for each state $s \in St(e)$, $s \models \psi$.

For the verification of invariance formulas, we also need to consider verification conditions of the form

$$[\varphi]\, p\, [\psi]$$

which states that if the agent starts executing p (a sequential plan) in a state satisfying φ then it will terminate successfully in a state satisfying ψ.

We say that a capability function $\mathcal{C}$ *satisfies* a formula $[\varphi]\, p\, [\psi]$, written $\mathcal{C} \models [\varphi]\, p\, [\psi]$, if each execution process $e \in \mathcal{C}(p, \varphi)$ is successful and terminates in a state satisfying ψ.

Verifying $[\varphi]\, p\, [\psi]$

To prove the condition $[\varphi]\, p\, [\psi]$, we need another kind of conditions of the form

$$\{\varphi\} \text{ p } \mathbf{inv}\ \gamma\ \{\psi\}$$

which states that every successful execution process $e \in Exe(p, \varphi)$ satisfying the property that γ is invariant during it (i.e. $\forall s \in St(e) : s \models \gamma$), terminates in a state satisfying ψ.

It is important to note that the validity of condition $\{\varphi\}$ p $\mathbf{inv}\ \gamma\ \{\psi\}$ does not depend on the agent's capabilities. They are therefore often referred to as *cap-free conditions.*

We can now give the rule for proving $[\varphi]\, p\, [\psi]$

- (SP)

$$\frac{\begin{array}{c}\textbf{executable}\ \text{p}\ \textbf{when}\ \varphi\ \textbf{before}\ \gamma \\ \{\varphi\}\, p\ \mathbf{inv}\ \neg\gamma\ \{\psi\}\end{array}}{[\varphi]\ \text{p}\ [\psi]}$$

Lemma 2 *The rule (SP) is sound in the sense that for each capability function $\mathcal{C}$, if $\mathcal{C} \models$ **executable** p **when** φ **before** γ, and $\{\varphi\}\, p$ **inv** $\neg\gamma$ $\{\psi\}$ holds then $\mathcal{C} \models [\varphi]p[\psi]$*

Proof Rules for Cap-Free Conditions

For each fluent proposition φ, define $T_A(\varphi) = \{T_A(s) \mid s \models \varphi\}$. Further we write $T_A(\varphi) \models \psi$ if each state in $T_A(\varphi)$ satisfies ψ.

The following notion of environment invariance plays an important role in the rules for proving cap-free conditions.

Definition 8 *1. A fluent proposition φ is said to be* environment invariant with respect to γ starting from ψ *if for each environment process e, if $initial(e) \models \psi$ and $\forall s \in St(e) : s \models \gamma$ then $\forall s \in St(e) : s \models \varphi$.*

2. We often say that φ is environment invariant starting from ψ if $\gamma \equiv true$

3. We also say that φ is environment invariant if it is environment invariant starting from φ.

We can now introduce the rules for proving cap-free conditions.

- (CF1)

$$\frac{\varphi \text{ is environment invariant wrt } \gamma \text{ starting from } \varphi \qquad T_A(\varphi \wedge \gamma) \models \gamma \rightarrow \psi}{\{\varphi\}\ \mathrm{A}\ \mathbf{inv}\ \gamma\ \{\psi\}}$$

- (CF2)

$$\frac{\{\varphi\}\, p\ \mathbf{inv}\ \gamma\ \{\phi\} \qquad \{\phi\}\, q\ \mathbf{inv}\ \gamma\ \{\psi\}}{\{\varphi\}\, p; q\ \mathbf{inv}\ \gamma\ \{\psi\}}$$

- (CF3)

$$\frac{\varphi' \rightarrow \varphi,\ \{\varphi\}\, p\ \mathbf{inv}\ \gamma\, \{\psi\},\ \psi \rightarrow \psi'}{\{\varphi'\}\, p\ \mathbf{inv}\ \gamma\, \{\psi'\}}$$

Lemma 3 *The proof system for cap-free conditions is sound.*

Proof Theory for Ability Propositions

Rules for Sequential Operator

- (CP1)

$$\frac{\mathbf{executable}\ \mathrm{p;q}\ \mathbf{when}\ \varphi\ \mathbf{before}\ \psi}{\mathbf{executable}\ \mathrm{p}\ \mathbf{when}\ \varphi\ \mathbf{before}\ \psi}$$

- (CP2)

$$\frac{[\varphi]\, p\, [\psi] \qquad \mathbf{executable}\ \mathrm{p}\ \mathbf{when}\ \varphi\ \mathbf{before}\ \phi \qquad \mathbf{executable}\ \mathrm{q}\ \mathbf{when}\ \psi\ \mathbf{before}\ \phi}{\mathbf{executable}\ \mathrm{p;q}\ \mathbf{when}\ \varphi\ \mathbf{before}\ \phi}$$

- (CP3)

$$\frac{\begin{array}{c}\mathbf{executable}\ p; A\ \mathbf{when}\ \varphi\ \mathbf{before}\ \psi \\ \mathbf{executable}\ p\ \mathbf{when}\ \varphi\ \mathbf{before}\ \gamma \\ [\varphi]\, p\, [\phi] \\ \{\phi\} A\ \mathbf{inv}\ \neg\psi[\neg\gamma] \\ \neg\gamma \text{ is environment invariant wrt } \neg\psi \text{ starting from } \phi\end{array}}{\mathbf{executable}\ p; A\ \mathbf{when}\ \varphi\ \mathbf{before}\ \gamma}$$

Rule for Atomic Actions

- (CP4)

$$\frac{\begin{array}{c}\mathbf{executable}\ \mathrm{A}\ \mathbf{when}\ \varphi\ \mathbf{before}\ \gamma \\ \{\varphi\}\, A\ \mathbf{inv}\ \neg\gamma\, \{\neg\psi\} \\ \neg\psi \text{ is environment invariant wrt } \neg\gamma \text{ starting from } \varphi\end{array}}{\mathbf{executable}\ \mathrm{A}\ \mathbf{when}\ \varphi\ \mathbf{before}\ \psi}$$

Other Rules

- (CP5)

$$\frac{\mathbf{executable}\ \mathrm{p}\ \mathbf{when}\ \varphi\ \mathbf{before}\ \psi \qquad \mathbf{executable}\ \mathrm{p}\ \mathbf{when}\ \varphi_1\ \mathbf{before}\ \psi_1}{\mathbf{executable}\ \mathrm{p}\ \mathbf{when}\ \varphi \wedge \varphi_1\ \mathbf{before}\ \psi \vee \psi_1}$$

- (CP6)

$$\frac{\mathbf{executable}\ \mathrm{p}\ \mathbf{when}\ \varphi\ \mathbf{before}\ \psi \qquad \mathbf{executable}\ \mathrm{p}\ \mathbf{when}\ \varphi_1\ \mathbf{before}\ \psi_1}{\mathbf{executable}\ \mathrm{p}\ \mathbf{when}\ \varphi \vee \varphi_1\ \mathbf{before}\ \psi \wedge \psi_1}$$

- (CP7)

$$\frac{\mathbf{executable}\ \mathrm{p}\ \mathbf{when}\ \varphi\ \mathbf{before}\ \psi \qquad \varphi_1 \models \varphi,\ \ \psi_1 \models \psi}{\mathbf{executable}\ \mathrm{p}\ \mathbf{when}\ \varphi_1\ \mathbf{before}\ \psi_1}$$

For each set of ability proposition $\mathcal{E}$, let $\Gamma_{\mathcal{E}}$ be the set of rules obtained by adding to the rules for ability propositions a new rule of the form

- (CP8)

$$\frac{\epsilon \in \mathcal{E}}{\epsilon}$$

Lemma 4 *The proof system for the ability propositions is sound in the sense that for each set of ability propositions $\mathcal{E}$, if $\mathcal{E} \vdash_{\Gamma_{\mathcal{E}}} \epsilon$, then $\mathcal{E} \models \epsilon$*

Rule for Invariance Formulae

We can now give the rule for invariance formula.

- (IF)

$$\frac{\begin{array}{ll}\exists\phi : & \varphi \rightarrow \phi,\ \ \phi \rightarrow \psi \\ & \phi \text{ is environment invariant wrt } \neg cond(\mathcal{R}) \\ & \text{starting from } \phi \\ \forall c \in \mathcal{R}: & [\phi \wedge test(c)]\ \text{body(c)}\ [\phi] \\ & \mathbf{executable}\ \text{body(c)}\ \mathbf{when}\ \phi \wedge test(c) \\ & \mathbf{before}\ \neg\psi\end{array}}{\varphi \rightarrow \Box(\mathcal{R}, \psi)}$$

Lemma 5 *The rule (IF) is sound in the sense that for each alert capability function $\mathcal{C}$, for each reactive plan $\mathcal{R}$, if there exists ϕ such that folowing conditions are satisfied:*

- *$\varphi \rightarrow \phi$ and $\phi \rightarrow \psi$ are valid and ϕ is environment invariant wrt $\neg cond(\mathcal{R})$ starting from ϕ*
- *For each $c \in \mathcal{R}$, $\mathcal{C} \models [\phi \wedge test(c)]\, body(c)\, [\phi]$ and $\mathcal{C} \models$ **executable** body(c) **when** $\phi \wedge test(c)$ **before** $\neg\psi$*

then $\mathcal{C} \models \varphi \rightarrow \Box(\mathcal{R}, \psi)$

Example 2

Let ϵ *be the capability proposition (1). Further let* $\mathcal{R}_0 = \{\neg cc \Rightarrow sc; ec\}$.

We want to prove: $\{\epsilon\} \vdash (al \wedge \neg on) \rightarrow \Box(\mathcal{R}_0, al)$

It is easy to see that $al \wedge \neg on$ *is environment invariant. Using rule (CF1) we get*

$$\vdash \{al \wedge \neg on\}\, sc\; \textbf{\textit{inv}}\; \neg cc\, \{al \wedge on\} \tag{3}$$

$$\vdash \{al \wedge on\}\, ec\; \textbf{\textit{inv}}\; \neg cc\, \{al \wedge \neg on\} \tag{4}$$

From (3,4) using rule (CF2) we get

$$\vdash \{al \wedge \neg on\}\, sc; ec\; \textbf{\textit{inv}}\; \neg cc\, \{al \wedge \neg on\} \tag{5}$$

Therefore, it is obvious from (CF3) and (5)

$$\vdash \{al \wedge \neg on \wedge \neg cc\}\, sc; ec\; \textbf{\textit{inv}}\; \neg cc\, \{al \wedge \neg on\} \tag{6}$$

Let $\mathcal{E} = \{\epsilon\}$*, it follows from (CP8)* $\mathcal{E} \vdash \epsilon$*. Using rule (CP1) we have*

$$\mathcal{E} \vdash\; \textbf{\textit{executable}}\; sc\; \textbf{\textit{when}}\; \neg cc \wedge al \wedge \neg on\; \textbf{\textit{before}}\; cc \tag{7}$$

Hence using (SP) and from (3,7), it follows:

$$\mathcal{E} \vdash [al \wedge \neg on \wedge \neg cc]\, sc\, [al \wedge on] \tag{8}$$

Using rule (CF3) and (3), we get

$$\vdash \{al \wedge \neg on \wedge \neg cc\}\, sc\; \textbf{\textit{inv}}\; \neg cc\, \{al \wedge on\} \tag{9}$$

Using (CP4) and (7,9), we get

$$\mathcal{E} \vdash\; \textbf{\textit{executable}}\; sc\; \textbf{\textit{when}}\; \neg cc \wedge al \wedge \neg on\; \textbf{\textit{before}}\; \neg al \tag{10}$$

Using (CF3), we get from (4)

$$\vdash \{al \wedge on\}\, ec\; \textbf{\textit{inv}}\; \neg cc\, \{al\} \tag{11}$$

Using (CP3), we get from (11,8,9), ϵ *and the fact that* al *is environemt invariant wrt* $\neg cc$ *from* $al \wedge on$ *:*

$$\mathcal{E} \vdash\; \textbf{\textit{executable}}\; sc; ec\; \textbf{\textit{when}}\; \neg cc \wedge al \wedge \neg on\; \textbf{\textit{before}}\; \neg al \tag{12}$$

Using rule (SP), we can derive from ϵ *and (6)*

$$\mathcal{E} \vdash [al \wedge \neg on \wedge \neg cc]\, sc; ec\, [al \wedge \neg on] \tag{13}$$

Using rule (IF), where $\varphi \equiv al \wedge \neg on$*,* $\phi \equiv \varphi$*, and* $\psi \equiv al$*, we can conclude from (12,13) and the fact that* $al \wedge \neg on$ *is environment invariant:*

$\mathcal{E} \vdash (al \wedge \neg on) \rightarrow \Box(\mathcal{R}_0, al)$

Discussion

Reasoning about complex actions in dynamic environment where environment actions are allowed to occur randomly has also been studied in the literature lately (Baral et al 1998, De Giacomo et al 1997). But these works consider only agents with a capability $\mathcal{C}$ satisfying $\mathcal{C}(p, s) = Exe(p, s)$ for each p and s. In other words, their agents constitute a subset of the class of agents considered in this paper.

For simplification and ease of understanding, we have made an assumption that our agents are alert. In many real world applications, this assumption could be an oversimplification. Hence it is important to study agents which are not alert in the future. Integrating the framework of (Saffioti et al 1995) with ours could be promising here.

Acknowledgement

We would like to thank Bob Kowalski, Franchesca Toni, Marek Sergot, Rob Miller, Murray Shanahan, Fariba Sadri, Ber (Yongyuth Permpoontanalarp), Paolo Mancarella, Antonio Brogi for their many very valuable suggestions. We are also very grateful to the three anonymous referees for their constructive and helpful suggestions. The paper is partially supported by the EC Keep in Touch grant, LPKRR.

References

Agre P.E., Chapman D., 1987, *Pengi: An Implementation of a Theory of activity*, Proc. of AAAI'87, pp 268-272

Brooks R., 1991, *Intelligence without Reasons* Proc. of IJCAI'91, pp 569-595

C. Baral and Son T.C. *Relating theories of actions and reactive control*, http://cs.utep.edu/chitta, 1998

Cohen P.R., Levesque H.J., 1990, *Intention is choice with commitment*, Artificial Intelligence 42, No 3, 213-261

De Giacomo G., Lesperance Y, Levesque H.J. *Reasoning about concurrent actions, prioritized interupts and exogenous actions i the situation calculus*, IJCAI-97.

M. Georgeff and A. Lansky, 1987 *Reactive reasoning and planning*, In AAAI 87.

M. Gelfond and V. Lifschitz., 1993, *Representing actions and change by logic programs*, Journal of Logic Programming, 17(2,3,4):301–323.

Kowalski R., Sadri F.,1996, *Towards a Unified Agent Architecture that Combines Rationality with Reactivity*, Proceedings of International Workshop on Logic in Databases, San Miniato, Italy, Springer Verlag.

Lesperance Y, Levesque H.J., 1995, *Indexical knowledge and robot action - a logical account*, Artificial Intelligence, Vol 73, Feb 1995, pp 117-148

Levesque H.J., Reiter R., Lesperance Y., Lin F., and Scherl R. *GOLOG: A Logic Programming Language for Dynamic Domains* Journal of LP, Vol 31, 1997

R. Reiter., 1996, *Natural actions, concurrency and continuous time in the situation calculus*, In L. Aiello, J. Doyle, and S. Shapiro, editors, KR 96, pages 2–13, 1996.

Rosenschein S.J, Kaelbling L.P., 1995, *A situated view of representation and control*, Artificial Intelligence, Vol 73, Feb 1995, pp 149-176

Saffioti A., Konolige K., Ruspini E. *A multivalued logic approach to integrating planning and control* Artificial Intelligence, Vol 76, 1995

Shoham Y., 1993, *Agent-oriented Programming*, Artificial Intelligence, Vol 60, Feb 1993, pp 51-92

Thomas S.R., 1994, *The PLACA agent programming language* Proc. of ECAI'94 Workshop (ATAL), LNAI 890, 1994