# AGENT PROGRAMMING LANGUAGE WITH INCOMPLETE KNOWLEDGE - AGENTSPEAK(I)

Duc Vo and Aditya Ghose
*Decision Systems Laboratory*
*School of Information Technology and Computer Science*
*University of Wollongong*
*NSW 2522, Australia*

Keywords: Agent programming language, AgentSpeak, BDI agent, default theory, incomplete knowledge, replanning

Abstract: This paper proposes an agent programming language called AgentSpeak(I). This new language allows agent programs (1) to effectively perform while having incomplete knowledge of the environment, (2) to detect no-longer possible goals and re-plan these goals correspondingly, and (3) to behave reactively to changes of environment. Specifically, AgentSpeak(I) uses default theory as agent belief theory, agent always act with preferred default extension at current time point (i.e. preference may changes over time). A belief change operator for default theory is also provided to assist agent program to update its belief theory. Like other BDI agent programming languages, AgentSpeak(I) uses semantics of transitional system. It appears that the language is well suited for intelligent applications and high level control robots, which are required to perform in highly dynamic environment.

## 1 INTRODUCTION

In modelling rational agents, modelling agent's attitudes as *Belief*, *Desire*, *Intention* (BDI agent) has been the best known approach. This model was first introduced by a philosopher Michael Bratman (Bratman, 1987) in 1987 and 1988. There has been number of formalizations and implementations of BDI Agents such as (Rao, 1996; Rao and Georgeff, 1995; Rao and Georgeff, 1991; Hindriks et al., 1999; Riemsdijk et al., 2003; Dastani et al., 2003; Wooldridge, 2000). Belief, desire, intention attitudes of a rational agent represents the information that the agent has about the environment, the motivation of what it wants to do, and finally the plans that the agent intends to execute to achieve its desired goals. These mental attitudes are critical for achieving optimal performance when deliberation is subject to resource bounds (Bratman, 1987).

Although, researchers have tackled most issues of BDI agents from logical systems (Rao and Georgeff, 1991; Wobcke, 2002; Wooldridge, 2000) to logic programming (D'Inverno and Luck, 1998; Rao, 1996), the issue of acting with incomplete knowledge about the environment has not been addressed in BDI agent programming languages. Rational agent is desired to work in a highly dynamic environment, while having incomplete knowledge about the environment. In literature, there has been much work to address the problem of incompleteness of belief theory such as (Reiter, 1980; Delgrande et al., 1994; Brewka and Eiter, 2000; Alferes et al., 1996; Meyer et al., 2001; Ghose and Goebel, 1998; MaynardReidII and Shoham, 1998; Alchourrón et al., 1985). Another problem normally faced in open-dynamic environment is to adapt with changes of environment both to react to changes and to adjust strategies to achieve previously adopted goals. Again, this problem has not been focused by available BDI agent programming languages.

In this paper, we propose a new agent programming language called AgentSpeak(I) which allows agent programs to effectively perform with incomplete knowledge about the environment, and to dynamicly adapt with changes of the environment but persistently commit to its goals.

This paper includes six sections. The next section describes an scenario of rescue robot where agent (RBot) needs to reason, act, and react with incomplete knowledge about its highly dynamic environment. This example will be use throughout the paper in the subsequent sections to demonstrate presented theory. Section three discusses definition, syntax, and properties of agent programs in AgentSpeak(I): agent belief theory; goals and triggers; plans; intention; and events. Section four defines operational semantics for



Vo D. and Ghose A. (2004).
AGENT PROGRAMMING LANGUAGE WITH INCOMPLETE KNOWLEDGE - AGENTSPEAK(I).
In *Proceedings of the Sixth International Conference on Enterprise Information Systems*, pages 356-363
DOI: 10.5220/0002629803560363

AgentSpeak(I). Section five compares AgentSpeak(I) with existing agent programming languages: AgentSpeak(L), 3APL, and ConGolog. Finally, conclusion and future research section will remark what has been done in the paper and proposes research directions from this work.

## 2 EXAMPLE

Let us consider a scenario of rescue robot (named RBot). RBot works in disaster site. RBot's duty is to rescue trapped person from a node to the safe node $A$. The condition of the site is dynamic and unsure. The only definite knowledge that RBot has about the area is the map. RBot can only travel from one node to another if the path between two nodes is clear. Because of the limitation of its sensor, RBot can only sense the conditions between its node and adjacent nodes.

RBot has knowledge of where its locate *at*, where there is a *trapped* human, *path* between nodes, if a node is *on_fire*, if it is *carry*ing a person, and finally its goal to *rescue* trapped human. Basic actions of RBot are *move* between node, *pick* a person up, and *release* a person. RBot can only move from one node to another node if there is a direct path between two nodes and this path is clear. RBot can only pick a person up if it is located at the node where the person is trapped. RBot can only release a carried person at node $A$. Table below defines predicate symbols for RBot.
$at(x)$: The robot is at node x
$clear(x, y)$: Path between nodes x and y is clear
$path(x, y)$: There is a path between nodes x and y
$trapped(p, x)$: There is a person p trapped at node x
$carry(p)$: The robot is carrying person p
$on\_fire(x)$: node x is on fire (i.e. danger for RBot)
$rescue(p, x)$: Goal to rescue person p at node x
$move(x, y)$: Move from node x to an adjacent node y on an available $path(x, y)$
$pick(p)$: Pick person p up
$release(p)$: Release carried person p

In such high-dynamic environment, to accomplish the rescuing task, RBot should be able to make default assumptions when reasoning during its execution, and RBot should also be able to adapt with changes of the environment by modifying its plans, intentions.

## 3 AGENT PROGRAMS

**Agent belief theory**

In this section, we introduce the agent belief theory. Agents usually have incomplete knowledge about the world requiring a suitably expressive belief representation language. We take the usual non-monotonic reasoning stance insisting on the ability to represent *defaults* on a means for dealing with this incompleteness. In the rest of this paper we explore the consequences of using default logic (Reiter, 1980) as the belief representative language. However, most of our results would hold had some other non-monotonic reasoning formalism had been used instead.

We augment the belief representation language in AgentSpeak (L) with default rules. If $p$ is a predicate symbol, $t_1, ..., t_n$ are terms, then an atom is of the form $p(t_1, ..., t_n)$ denoted $p(\vec{t})$ (e.g. $at(x)$, $rescue(p, x)$, $clear(A, B)$). If $b_1(\vec{t_1})$ and $b_2(\vec{t_2})$ are belief atoms, then $b_1(\vec{t_1}) \wedge b_2(\vec{t_2})$ and $\neg b_1(\vec{t_1})$ are beliefs. A set $S$ of beliefs is said to be consistent iff $S$ does not have both a belief $b$ and its negation $\neg b$. A belief is called ground iff all terms in the belief are ground terms (e.g. $at(A)$, $trapped(P, F)$, $path(C, F) \wedge clear(C, F)$).

Let $\alpha(\vec{x})$, $\beta(\vec{x})$, and $\omega(\vec{x})$ be beliefs. A default is of the form $\frac{\alpha(\vec{x}):\beta(\vec{x})}{\omega(\vec{x})}$ where $\alpha(\vec{x})$ is called the *prerequisite*; $\beta(\vec{x})$ is called the *justification*; and $\omega(\vec{x})$ is called the *consequent* of the default (Reiter, 1980).

Interpretation of a default depends on variants of default logics being used (note that several exist (Reiter, 1980; Delgrande et al., 1994; Brewka and Eiter, 2000; Giordano and Martelli, 1994) exploring different intuitions). Informally, a default is interpreted as follows: "If for some set of ground terms $\vec{c}$, $\alpha(\vec{c})$ is provable from what is known and $\beta(\vec{c})$ is consistent, then conclude by default that $\omega(\vec{c})$".

A *default theory* is a pair $(W, D)$, where $W$ is a consistent set of beliefs and $D$ is a set of default rules. Initial default theory of RBot is presented in table 1.

**Example 1** *Initial default theory* $\Delta_{RBot} = (W_{RBot}, D_{RBot})$ *of RBot*

$$W_{RBot} = \left\{ \begin{array}{ll} path(A, B), & path(B, C), \\ path(C, D), & path(C, E), \\ path(C, F), & path(D, F), \\ path(E, F), & at(x) \wedge at(y) \rightarrow \\ x = y & \end{array} \right\}$$

$$D_{RBot} = \left\{ \begin{array}{ll} \frac{:at(A}{at(A)}, & \frac{:clear(A,B}{clear(A,B)}, \\ \frac{:\neg carry(p)}{\neg carry(p)}, & \frac{:trapped(P,F)}{trapped(P,F)}, \\ \frac{path(x,y):clear(x,y)}{clear(x,y)}, & \\ \frac{:\neg trapped(p,y)}{\neg trapped(p,y)}, & \frac{:\neg path(x,y)}{\neg path(x,y)}, \\ \frac{:\neg clear(x,y)}{\neg clear(x,y)}, & \end{array} \right\}$$

Much of out discussion is independent of the specific variant being used. We only require that at least an extension must exist. This is not generally true for Reiter's default logic (Reiter, 1980). However, if one restricts attention to semi- normal default theories, there is a guarantee that an extension will always

exist. We make this assumption here. i.e belief theories must necessarily only be semi-normal default theories.

**Example 2** *With Reiter semantics, default extensions of* $\Delta_{RBot}$ *would be*

$E_{RBot1} = Cn(\{ at(A), path(A,B), clear(A,B), path(B,C), path(C,F), \neg carry(p), trapped(P,F), clear(B,C), \neg clear(B,D), \neg clear(D,F), clear(C,F), ...\})$'

$E_{RBot2} = Cn(\{ at(A), path(A,B), clear(A,B), path(B,C), path(C,F), \neg carry(p), trapped(P,F), clear(B,D), clear(D,F), \neg clear(C,F), ...\})$

*etc.*

To operate, an agent program needs to commit with one extension of its default theory. There is an extension selection function for agent to select most preferred extension from set of extensions of its default theory for further execution. Let $S_{\mathcal{E}}$ be a extension selection function, if $B = S_{\mathcal{E}}(\Delta)$, then (1) $B$ is a default extension of $\Delta$ and (2) $B$ is the most preferred extension by the agent at the time where $S_{\mathcal{E}}$ is applied. In the rest of this paper, the *current agent belief set* will be denoted by $B = S_{\mathcal{E}}(\Delta)$ given an agent belief theory $\mathcal{B} = \langle \Delta, S_{\mathcal{E}} \rangle$.

**Belief Change Operators**

Belief change is a complicated issue. There have been several well known work on belief change such as (Alchourrón et al., 1985; Ghose et al., 1998; Meyer et al., 2001; Darwiche and Pearl, 1997; Ghose and Goebel, 1998). In this paper, we do not discuss this issue in detail. However for the completeness of out system, we adopt belief change framework of (Ghose et al., 1998). We denote $\circ_g$ (respectively $-_g$) as Ghose's revision (respectively contraction) operator.

When updating agent belief theory, we assumes that (1) the belief to be revised must be a consistent belief, (2) the belief to be revised must be consistent with the set of the base facts of belief theory, (3) the belief to be contracted must not be a tautology and (4) the belief to be contracted must not be entailed by the base facts.

**Goals, Triggers, Plans and Intentions**

We follow the original definitions in (Rao, 1996) to define goals and triggering events. Two types of goals are of interest: *achievement goals* and *test goals*. An achievement goal, denoted $!g(\vec{t})$, indicates an agent's desire to achieve a state of affairs in which $g(\vec{t})$ is true. A test goal, denoted $?g(\vec{t})$, indicates an agent's desire to determine if $g(\vec{t})$ is true relative to its current beliefs. Test goals are typically used to identify unifiers that make the test goal true, which are then used to instantiate the rest of the plan. If $b(\vec{t})$ is a belief and $!g(\vec{t})$ is an achievement goal, then $+b(\vec{t})$ (add a belief $b(\vec{t})$), $-b(\vec{t})$ (remove a belief $b(\vec{t})$), $+!g(\vec{t})$ (add an achievement goal $!g(\vec{t})$), and $-!g(\vec{t})$ (remove the achievement goal $g(\vec{t})$) are *triggering events*.

An agent program includes a *plan library*. The original AgentSpeak (Rao, 1996) definition views a *plan* as a triple consisting of a *trigger*, a *context* (a set of pre-conditions that must be entailed by the current set of beliefs) and a *body* (consisting of a sequence of atomic actions and sub-goals). We extend this notion to distinguish between an *invocation context* (the pre-conditions that must hold at the time that the plan in invoked) and an *invariant context* (conditions that must hold both at the time of plan invocation and at the invocation of every plan to achieve sub-goals in the body of the plan and their sub-goals). We view both kinds of contexts to involve both *hard pre-conditions* (sentences that must be true relative to the current set of beliefs) and *soft pre- conditions* (sentences which must be consistent with the current set of beliefs). Soft pre-conditions are akin to *assumptions*, *justifications* in default rules (Reiter, 1980) or *constraints* in hypothetical reasoning systems (Poole, 1988).

**Definition 1** *A* plan *is a 4-tuple* $\langle \tau, \chi, \chi^*, \pi \rangle$ *where* $\tau$ *is a trigger,* $\chi$ *is the invocation context,* $\chi^*$ *is the invariant context and* $\pi$ *is the body of the plan. Both* $\chi$ *and* $\chi^*$ *are pairs of the form* $(\beta, \alpha)$ *where* $\beta$ *denotes the set of hard pre-conditions while* $\alpha$ *denotes the set of soft pre-conditions. A plan* $p$ *is written as* $\langle \tau, \chi, \chi^*, \pi \rangle$ *where* $\chi = (\beta, \alpha)$ *(also referred to as* $InvocationContext(p)$*),* $\chi^* = (\beta^*, \alpha^*)$ *(also referred to as* $InvariantContext(p)$*),* $\pi =< h_1, \ldots, h_n >$ *(also referred to as* $Body(p)$*) and each* $h_i$ *is either an atomic action or a goal. We will also use* $Trigger(p)$ *to refer to the trigger* $\tau$ *of plan* $p$*.*

**Example 3** *RBot's plan library:*

$p_1 = \langle +!at(y), (\{at(x)\}, \{\emptyset\}), (\{\emptyset\}, \{clear(x,y)\}), \langle move(x,y) \rangle \rangle$

$p_2 = \langle +!at(y), (\{\neg at(x), path(x,y)\}, \{\emptyset\}), (\{\emptyset\}, \{clear(x,y)\}), \langle !at(x), ?clear(x,y), move(x,y) \rangle \rangle$

$p_3 = \langle +!rescue(p,x), (\{\emptyset\}, \{\emptyset\}), (\{\emptyset\}, \{trapped(p,x) \vee carry(p)\}), \langle !at(x), pick(p), !at(A), release(p) \rangle \rangle$

$p_4 = \langle +on\_fire(x), (\{at(x), \neg on\_fire(y), path(x,y)\}, \{clear(x,y)\}), (\{\emptyset\}, \{\emptyset\}), \langle move(x,y) \rangle \rangle$

$p_5 = \langle +trapped(p,x), (\{\emptyset\}, \{\emptyset\}), (\{\emptyset\}, \{\emptyset\}), \langle !rescue(p,x) \rangle \rangle$

$P_{RBot} = \{p_1, p_2, p_3, p_4, p_5\}$

In example 3, plan $p_1$ and $p_2$ are RBot strategies to get to a specific node on the map. Plan $p_3$ is the strategy assisting RBot to decide how to rescue a person trapped in a node. Plan $p_4$ is a reactive-plan for RBot to get out of on-fire node. Plan $p_5$ is another reactive-plan for RBot to try rescue a person, when RBot adds a new belief that person is trapped at some node.

As with (Rao, 1996), a plan $p$ is deemed to be a relevant plan relative to a triggering event $\tau$ if and only if there exists a most general unifier $\sigma$ such that $d\sigma = \tau\sigma$, where $Trigger(p) = \tau$. $\sigma$ if referred to as the *relevant unifier* for $p$ given $\tau$.

**Definition 2** *A plan $p$ of the form $\langle \tau, \chi, \chi^*, \pi \rangle$ is deemed to be an* applicable plan *relative to a triggering event $\tau'$ and a current belief set $B$ iff:*

*(1) There exists a relevant unifier $\sigma$ for $p$ given $\tau'$.*

*(2) There exists a substitution $\theta$ such that $\beta\sigma\theta \cup \beta^*\sigma\theta \subseteq Th(B)$*

*(3) $\alpha\sigma\theta \cup \alpha^*\sigma\theta \cup B$ is satisfiable.*

*$\sigma\theta$ is referred to as the* applicable unifier *for $\tau'$ and $\theta$ is called its* correct answer substitution.

Thus, a relevant plan is applicable if the hard pre- conditions (both in the invocation and invariant contexts) are entailed by its current set of beliefs and the soft pre- conditions (both in the invocation and invariant contexts) are consistent with its current set of beliefs.

**Example 4** *Plan $p_3$ is intended by trigger $+!rescue(P, F)$ and agent belief set $E_1$ with correct answer substitution $\{p/P, x/F\}$*

$(p_3)\sigma\{p/P, x/F\} =$
$\langle +!rescue(P, F)$, $(\{\emptyset\}, \{\emptyset\})$, $(\{\emptyset\}, \{trapped(P, F) \vee carry(p)\})$, $\langle !at(F)$, $pick(P)$, $!at(A)$, $release(P)$ $\rangle\rangle$

**Definition 3** *An intended (partially instantiated) plan $\langle \tau, \chi, \chi^*, \pi \rangle$ $\sigma\theta$ is deemed to be executable with respect to belief set $B$ iff*

*(1) $\beta^*\sigma\theta \subseteq Th(B)$*

*(2) $\alpha^*\sigma\theta \cup B$ is satisfiable*

**Example 5** *Plan $(p_3)\sigma\{p/P, x/F\}$ is executable with respect to belief set $E_2$.*

Syntactically, our plan is not much different to the one in (Rao, 1996), however, the partition of context to four parts will give the agent more flexible when applying and executing a plan. This way of presenting a plan also provides agent's ability to discover when thing goes wrong or turns out not as expected (i.e. when invariant context is violated).

An intention is a state of intending to act that a rational agent is going to do to achieve its goal (Bratman, 1987).

Formally, an intention is defined as

**Definition 4** *Let $p_1, ..., p_n$ be partially instantiated plans (i.e. instantiate by some applicable substitution), an intention $\iota$ is a pre- ordered tuple $\langle p_1, ..., p_n \rangle$. Where*

*(1) $Trg_P(p_1)$ is called original trigger of $\iota$, denote $Trg_I(\iota) = Trg_P(p_1)$*

*(2) An intention $\iota = \langle p_1, ..., p_n \rangle$ is said to be valid with respect to current belief set $B$ iff $\forall i$ $p_i$ is executable with respect to $B$ (definition 3).*

*(3) An intention is said to be true intention if it is of the form $\langle\rangle$ (empty). A true intention is always valid.*

*(4) An intention $\iota = \langle p_1, ..., p_n \rangle$ is said to be invalid with respect to current belief set $B$ if it is not valid.*

*(5) Another way of writing an intention $\iota$ is $\langle \iota', p_n \rangle$ with $\iota' = \langle p_1, ..., p_{n-1} \rangle$ is also an intention.*

**Example 6** *At node $A$, RBot may have an intention to rescue a trapped person at node $F$ as*

$$\iota_1 = \langle p_3\sigma\{p/P, x/F\} \rangle$$

**Events**

We adopt notion of agent events in AgentSpeak(L) (Rao, 1996). An events is a special attribute of agent internal state, an event can be either external (i.e. events are originated by environment e.g. users or other agents) or internal (i.e. evens are originated by internal processes of agent).

**Definition 5** *Let $\tau$ be a ground trigger, let $\iota$ be an intention, an event is a pair $\langle \tau, \iota \rangle$.*

*(1) An event is call external event if its intention is a true intention, otherwise it is called internal event.*

*(2) An event is valid with respect to agent current belief set $B$ if its intention is valid with respect to $B$, otherwise it is invalid event.*

*(3) Let $e = \langle \tau, \iota \rangle$ be an event, denote*

$$Trg_E(e) = \left\{ \begin{array}{ll} \tau & \text{if e is external} \\ Trg_I(\iota) & \text{if e is internal} \end{array} \right\}$$

**Example 7** *External Event:*
$e_1 = \langle +!rescue(P, F), \langle\rangle \rangle$
*Internal Event:*
$e_2 = \langle +!at(F), \langle p_3' \rangle \rangle$
*where*
$p_3'$=$\langle +!rescue(P, F)$, $(\{\emptyset\}, \{\emptyset\})$, $(\{\emptyset\}, \{trapped(P, F) \vee carry(p)\})$, $\langle pick(P)$, $!at(A)$, $release(P)$ $\rangle\rangle$

**Corollary 1** *All external events of an agent are valid in respects of its current belief set.*

# 4 OPERATIONAL SEMANTICS

Like other BDI agent programming languages (Rao, 1996; Hindriks et al., 1999; Levesque et al., 1997), we use transitional semantics for our system.

Informally, an agent program in AgentSpeak(I) consists of a belief theory $\mathcal{B}$, a set of events $E$, a set of intention $I$, a plan library $P$, and three selection functions $\mathcal{S}_E$, $\mathcal{S}_P$, $\mathcal{S}_I$ to select an event, a plan, an intention (respectively) to process.

**Definition 6** *An agent program is a tuple*

$$\langle \mathcal{B}, P, E, I, \mathcal{S}_P, \mathcal{S}_E, \mathcal{S}_I \rangle$$

*where*

*(1)* $\mathcal{B} = \langle \Delta, S_\mathcal{E} \rangle$ *is agent belief theory of the agent.*
*(2) At any time denote* $B = S_\mathcal{E}(\Delta)$ *is current belief set of the agent.*
*(3)* $E$ *is set of events (including external events and internal events).*
*(4)* $P$ *is agent plan repository, a library of agent plans.*
*(5)* $I$ *is a set of intentions.*
*(6)* $\mathcal{S}_E$ *is a selection function which selects an event to process from set* $E$ *of events.*
*(7)* $\mathcal{S}_P$ *is a selection function which selects an applicable plan to a trigger* $\tau$ *from set* $P$ *of plans.*
*(8)* $\mathcal{S}_I$ *is a selection function which selects an intention to execute from set* $I$ *of intentions.*
*(9)* $\mathcal{S}_E$*/*$\mathcal{S}_P$*/*$\mathcal{S}_I$ *returns null value if it fails to select an extension/intention/plan*

**Example 8** *The initial agent program of RBot at node a would be like*
$\langle \mathcal{B}_{RBot}, P_{RBot}, \{\langle +!rescue(person, f), \langle\rangle\rangle\}, \langle\rangle,$
$\mathcal{S}_E, \mathcal{S}_P, \mathcal{S}_I \rangle$
*Where* $\mathcal{S}_E$*,* $\mathcal{S}_P$*,* $\mathcal{S}_I$ *are some valid selection functions (e.g. select the first valid option).*

There are two types of transits in AgentSpeak(I): *Transit to process events* and *Transit to execute intention*. These transitions may run in sequent or in parallel. The choice of using which method depends on specific domain.

As shown in definitions 4 and 5, an agent intention or an event can be sometime invalid. A good agent program should appropriately response to such cases. We propose two functions $Repair_I$ and $Val_E$ for the purposes repairing invalid intentions and validate events when executing AgentSpeak(I) agent programs. The $Repair_I$ function takes an invalid intention as its input and outputs an event which is then valid with respect to agent current belief set. The $Val_E$ function is slightly different. It takes an event as its input and outputs an valid event.

**Definition 7** *Let* $\mathbb{I}$ *be set of all intentions,* $\mathbb{E}$ *be set of all events,* $\mathbb{B}$ *be set of all belief sets. Let* $Repair_I : \mathbb{I}, \mathbb{B} \rightarrow \mathbb{E}$ *be a repairing function which modify an* invalid intention $\iota$ *with respect to belief set* $B$ *to return an* valid event $\epsilon$ *with respect to belief set* $B$.

The function $Rep_i$ must satisfy following conditions

RI-1. if $\langle \tau, \iota' \rangle = Rep_i(\iota, B)$, then (1) $\iota'$ and $\iota$ are in the forms of $\langle p_1, ..., p_k \rangle$ and $\langle p_1, ..., p_k, p_{k+1}, ...p_n \rangle$ respectively where $(k < n)$, and (2) $\iota'$ is valid with respect to belief set $B$.

RI-2. if if $\langle \tau, \iota' \rangle = Rep_i(\iota, B)$, where $\iota' = \langle p_1, ..., p_k \rangle$ and $\iota = \langle p_1, ..., p_k, p_{k+1}...p_n \rangle$, then $\tau = Trg_P(p_{k+1})$.

**Definition 8** *Let* $\mathbb{E}$ *be set of all events,* $\mathbb{B}$ *be set of all belief sets. If* $B$ *is current agent belief set,* $e = \langle \tau, \iota \rangle$ *is an event, then function* $Val_E :: \mathbb{E}, \mathbb{B} \rightarrow \mathbb{E}$ *is defined as*

$$Val_E(e, B) = \left\{ \begin{array}{ll} e & \text{if } e \text{ is valid wrt } B \\ Rep_i(\iota, B) & \text{if } e \text{ is invalid wrt } B \end{array} \right\}$$

When an event $e = \langle \tau, \iota \rangle$ is selected by event selection function $\mathcal{S}_E$. $Val_E$ function helps to make sure that $e$ valid with respect to agent current belief set $B$, and ready to be processed.

**Event Processing**

Transition of agent program to process events depends on the event triggers. An event trigger can be

1. Add an achievement goal $+!g(\vec{t})$
2. Remove an achievement goal $-!g(\vec{t})$
3. Add a belief $+b$
4. Remove a belief $-b$

In case of adding an achievement goal $+!g(\vec{t})$, our agent (1) selects an applicable plan $p$ from the plan library, (2) partially instantiate $p$ with applicable substitution $\theta$ by unifier $\sigma$, (3) appends $p\sigma\theta$ to intention part of the event, and (4) adds that intention into the agent's intention set.

**Definition 9** *Let* $Val_E(\mathcal{S}_E(E)) = \langle +!g(\vec{t}), \iota \rangle$*, let* $\mathcal{S}_P(P) = p$*, and where partially instantiated plan* $p$ *is an applicable plan relative to triggering event* $+!g(\vec{t})$ *and current belief set* $B$*.* $e$ *is said to be processed iff* $\langle \iota, p \rangle \in I$

In case of removing an achievement goal $-!g(\vec{t})$ (remind that $-g(\vec{t})$ is grounded), our agent (1) removes any plans which are triggered by ground trigger $+!g(\vec{t})$ from agent sets of intentions and events, (2) removes any sub-plans of those plans (i.e. plans which have higher priorities in the same intention).

**Definition 10** *Let* $Val_E(\mathcal{S}_E(E)) = e$ *where* $e = \langle -!g(\vec{t}), \iota \rangle$*.* $e$ *is said to be processed iff for any intention*

$$\iota \in I \cup \{\iota' | \langle \tau, \iota' \rangle \in E\}$$

*if (1)* $\iota$ *is in the form of* $\langle p_1, ..., p_k, ..., p_n \rangle$*, (2)* $p_k = \langle +!g(\vec{t}), \chi, \chi^*, \pi \rangle$ *and (3) for any* $i < k$ $p_i = \langle \tau_i, \chi_i, \chi_i^*, \pi_i \rangle$ $\tau$ *does not unify with* $+!g(\vec{t})$*, then* $\iota$ *is cut to be* $\langle p_1, ..., p_{k-1}$ *(note that* $\iota$ *may become an empty intention).*

In case of adding a belief, the agent revises its agent belief theory with this belief and adopts an applicable plan from its plan library to react to this change of its beliefs (reactive characteristic).

**Definition 11** *Let $Val_E(\mathcal{S}_E(E)) = e$ where $e = \langle +b(\vec{t}), \iota \rangle$, let $\mathcal{S}_P(P) = p$ where partially instantiated $p$ is an applicable plan relative to triggering event $+!g(\vec{t})$ and current belief set $B$. $e$ is said to be processed iff (1) $\mathcal{B} = \mathcal{B} \circ_G b$ and (2) $\langle \iota, p \rangle \in I$*

Finally, in case of removing a belief, the agent contracts that belief from its agent belief theory and adopts an applicable plan from its plan library to react to this change of its beliefs (reactive characteristic).

**Definition 12** *Let $Val_E(\mathcal{S}_E(E)) = e$ where $e = \langle -b(\vec{t}), \iota \rangle$, let $\mathcal{S}_P(P) = p$ where partially instantiated $p$ is an applicable plan relative to triggering event $+!g(\vec{t})$ and current belief set $B$. $e$ is said to be processed iff (1) $\mathcal{B} = \mathcal{B} -_G b$ and (2) $\langle \iota, p \rangle \in I$*

We have following algorithm for *event processing*

1: $e = \mathcal{S}_E(E)$
2: **if** $e$ is not null **then**
3: $E = E \setminus \{e\}$
4: $B = S_{\mathcal{E}}(\Delta)$
5: $\langle \tau, \iota \rangle = Val_E(e, B)$
6: **if** $\tau$ is of the form $+!g(\vec{t})$ **then**
7: $p = \mathcal{S}_P(\tau, B, P)$
8: **if** $p$ is null **then**
9: $E = E \cup \{\langle \tau, \iota \rangle\}$
10: **else**
11: $I = I \cup \{\langle \iota, p \rangle\}$
12: **else if** $\tau$ is of the form $-!g(\vec{t})$ **then**
13: **for all** $\iota \in I$ **do**
14: **if** $\iota = \langle p_1, ..., p_k, \langle +!g(\vec{t}), ... \rangle, ..., p_n \rangle$ **then**
15: $\iota = \langle p_1, ..., p_k \rangle$
16: **for all** $\langle \tau, \iota \rangle \in E$ **do**
17: **if** $\iota = \langle p_1, ..., p_k, \langle +!g(\vec{t}), ... \rangle, ..., p_n \rangle$ **then**
18: $\iota' = \langle p_1, ..., p_{k-1} \rangle$
19: $E = E \setminus \langle \tau, \iota \rangle \cup \langle trigger(p_k), \iota' \rangle$
20: **else if** $\tau$ is of the form $+b(\vec{t})$ **then**
21: $\Delta = \Delta \circ b(\vec{t})$
22: $p = \mathcal{S}_P(\tau, B, P)$
23: **if** $p$ is null **then**
24: $I = I \cup \{\iota\}$
25: **else**
26: $I = I \cup \{\langle \iota, p \rangle\}$
27: **else if** $\tau$ is of the form $-b(\vec{t})$ **then**
28: $\Delta = \Delta - b(\vec{t})$
29: $p = \mathcal{S}_P(\tau, B, P)$
30: **if** $p$ is null **then**
31: $I = I \cup \{\iota\}$
32: **else**
33: $I = I \cup \{\langle \iota, p \rangle\}$

In our example, at node $a$ there is only one external event for the robot to select, which is $+!rescue(P, F)$ to rescue $P$ at node $F$. There is also only one plan intended with this event and selected belief set $E_2$, which is $p_3$ instantiated by substitution $\{P/p, F/x\}$. Hence, an intention will be generated to put into set of intentions, which now is $\langle +!rescue(P, F), (\{\emptyset\}, \{\emptyset\}), (\{\emptyset\}, \{trapped(P, F) \vee carry(p)\}), (!at(F), pick(P), !at(A), release(P)) \rangle\rangle$

After this execution, the set of event is an empty set.

**Intention Executing**

Executing an intention is an important transition of AgentSpeak(I) agent program, it is the process when the agent acts pro-actively towards its environment to achieve some environmental stage. An intention will be executed based on the first formula in the body of the highest priority plan of the intention. This formula can be: an achievement goal, a test goal, or an action. In case of an achievement goal, an internal event will be generated. In case of a test goal, the agent will test if there exists a set of ground terms $\vec{c}$ that makes the test goal an element of the current belief set, and use this set for subsequential execution of the intention. Finally, in case of an action, the agent will perform that action, which may result in changing of environmental state. These executions are formalized below.

**Definition 13** *Let $\mathcal{S}_I(I) = \iota$, where*

$$\iota = \langle \iota', \langle \tau, \chi, \chi^*, \langle !g(\vec{t}), h_2, ..., h_n \rangle \rangle \rangle$$

*The intention $\iota$ is said to be executed iff*

$$\langle +!g(\vec{t}), \langle \iota', \langle \tau, \chi, \chi^*, \langle h_2, ..., h_n \rangle \rangle \rangle \rangle \in E$$

**Definition 14** *Let $\mathcal{S}_I(I) = \iota$, where*

$$\iota = \langle \iota', \langle \tau, \chi, \chi^*, \langle ?g(\vec{t}), h_2, ..., h_n \rangle \rangle \rangle$$

*The intention $\iota$ is said to be executed iff*

*(1) there exists an substitution $\theta$ such that $g(\vec{t})\theta \in B$ and*

*(2) $\iota$ is replaced by*

$$\iota = \langle \iota', \langle \tau, \chi, \chi^*, \langle h_2\theta, ..., h_n\theta \rangle \rangle \rangle$$

**Definition 15** *Let $\mathcal{S}_I(I) = \iota$, where*

$$\iota = \langle \iota', \langle \tau, \chi, \chi^*, \langle a(\vec{t}), h_2, ..., h_n \rangle \rangle \rangle$$

*The intention $\iota$ is said to be executed iff (1) $\iota$ is replaced by*

$$\iota = \langle \iota', \langle \tau, \chi, \chi^*, \langle h_2, ..., h_n \rangle \rangle \rangle$$

*and (2) $a(\vec{t})$ is sent to agent action processor.*

**Definition 16** *Let $\mathcal{S}_I(I) = \iota$, where*

$$\iota = \langle \iota', \langle \tau, \chi, \chi^*, \langle \rangle \rangle \rangle$$

*The intention $\iota$ is said to be executed iff $\iota$ is replaced by $\iota'$*

Like processing an event, before executing any intention, the agent need to verify if the intention is still valid in its current belief set $B$, and repairs that intention if necessary.

We have following algorithm for *intention executing*

1: $\iota = \mathcal{S}_I(I)$
2: **if** $\iota$ is not null **then**
3: $I = I \setminus \{\iota\}$
4: $B = S_{\mathcal{E}}(\Delta)$
5: **if** $\iota$ is invalid with respect to $B$ **then**
6: $e = Rep_i(\iota)$
7: $E = E \cup \{e\}$
8: **else**
9: present $\iota$ as $\langle \iota', p \rangle$
10: **if** $body(p)$ is empty **then**
11: $I = I \cup \{\iota'\}$
12: **else**
13: present $p$ as $\langle \tau, \chi, \chi^*, \langle h, h_2, ..., h_n \rangle \rangle$
14: **if** $h$ is an action **then**
15: perform $h$
16: $p' = \langle \tau, \chi, \chi^*, \langle h_2, ..., h_n \rangle \rangle$
17: $I = I \cup \{\langle \iota', p' \rangle\}$
18: **else if** $h$ is of the form $!g(\vec{t})$ **then**
19: $p' = \langle \tau, \chi, \chi^*, \langle h_2, ..., h_n \rangle \rangle$
20: $e = \langle +h, \langle \iota', p' \rangle \rangle$
21: $E = E \cup \{e\}$
22: **else if** $h$ is of the form $?g(\vec{t})$ **then**
23: find a substitution $\theta$ s.t. $g(\vec{t})\theta \in B$
24: **if** no substitution is found **then**
25: $I = I \cup \{\iota\}$
26: **else**
27: $p' = \langle \tau, \chi, \chi^*, \langle h_2\theta, ..., h_n\theta \rangle \rangle$
28: $I = I \cup \{\langle \iota', p' \rangle\}$

Again, in RBot mind now, there is only one intention to execute. This intention is currently valid. The first element of the intention is a goal $!at(f)$. Hence, an internal event is generated, which is $\langle +!at(F), \langle p'_3 \rangle \rangle$
where
$p'_3$=$\langle +!rescue(P,F), (\{\emptyset\}, \{\emptyset\}), (\{\emptyset\}, \{trapped(P,F) \vee carry(p)\}), \langle pick(P), !at(A), release(P) \rangle \rangle$

This event then is added into set of events. The original intention is removed from set of intentions.

# 5 COMPARISON WITH OTHER LANGUAGES

There have been several well known agent programming languages from very first language like Agent0 in 1993 (Shoham, 1993), to AgentSpeak(L) (Rao, 1996), Golog/ConGolog (Levesque et al., 1997; de Giacomo et al., 2000), 3APL (Hindriks et al., 1999) in late 1990s. In this section, we will compare AgentSpeak(I) with three later agent programming languages (AgentSpeak(L) , ConGolog, and 3APL). The extensive advantage of AgentSpeak(I) in comparing with these languages is that AgentSpeak(I) allows agents to act with incomplete knowledge about environment. Furthermore, others agent programming languages leave a gap in how agents propagating their own beliefs during agents life, which is reasonably covered in AgentSpeak(I).

**AgentSpeak(L)**: AgentSpeak(L) aims to implement of BDI agents in a logic programming style. It is an attempt to bridge the gap between logical theories of BDI and implementations. Syntactically, AgentSpeak(L) and AgentSpeak(I) are similar. However, the differences in plans, belief theory, and the semantics make AgentSpeak(I) programs extensively more capable than AgentSpeak(L) programs, especially when acting with incomplete knowledge about the environment.

**3APL**: 3APL is a rule-based language which is similar to AgentSpeak(I). A configuration of 3APL agent consists of a belief base, a goal base, and a set of practical reasoning (PR) rules, which are likely corresponding to belief theory, set of events and intentions, and plan library in AgentSpeak(I) respectively. The advantage that 3APL has over AgentSpeak(I) is the supporting of compound goals. Classification of PR rules in 3APL is only a special way of patition AgentSpeak(I) plan library, where we consider all plan with equal priority. Nevertheless, AgentSpeak(I) provides stronger support to agent abilities to perform in highly dynamic environment.

**ConGolog**: ConGolog is an extension of situation calculus. It is a concurrent language for high-level agent programming. ConGolog uses formal model semantics. ConGolog provides a logical perspective on agent programming in comparison with AgentSpeak(I) providing operational semantics that show how agent program propagates its internal states of beliefs, intentions, and events. Agent programs in ConGolog plan its actions from initial point of execution to achieve a goal. An assumption in ConGolog is that the environment changes only if a agent performs an action. This strong assumption is not required in AgentSpeak(I). Hence, ConGolog provides weeker supports to agent performance in comperation with AgentSpeak(I) when dealing with incomplete knowledge about the environment.

# 6 CONCLUSION & FUTURE RESEARCH

We have introduced a new agent programming language to deal with incomplete knowledge about envi-

ronment. Syntax and operational semantics of the languages has been presented. Comparison of AgentSpeak(I) and current agent programming languages has been discussed. In short, agent programs in AgentSpeak(I) can effectively perform in a highly dynamic environment by making assumptions at two levels: when computing belief set and when planning or replanning; detect planning problems raised by changes of the environment and re-plan when necessary at execution time; and finally propageting internal beliefs during execution time.

There are several directions that would be extended from this work. First, there would be an extension of background belief theory on belief change operators and temporal reasoning with beliefs and actions. Second, there would be an extension of this framework with multi-agent environment, where agent's intentions are influenced by its beliefs of others' beliefs and intentions. Third, there would be work on declarative goals on extension of AgentSpeak(I). And finally, there would be an extension which uses action theory to update agent beliefs during execution time.

## REFERENCES


Alchourrón, C. E., Gärdenfors, P., and Makinson, D. (1985). On the logic of theory change: Partial meet contraction and revision functions. *Journal of Symbolic Logic*, 50:510–530.

Alferes, J. J., Pereira, L. M., and Przymusinski, T. C. (1996). Belief revision in non-monotonic reasoning and logic programming. *Fundamenta Informaticae*, 28(1-2):1–22.

Bratman, M. E. (1987). *Intentions, Plans, and Practical Reason*. Harvard University Press, Cambridge, MA.

Brewka, G. and Eiter, T. (2000). Prioritizing default logic. *Intellectics and Computational Logic, Papers in Honor of Wolfgang Bibel, Kluwer Academic Publishers, Applied Logic Series*, 19:27–45.

Darwiche, A. and Pearl, J. (1997). On the logic of iterated belief revision. *Artificial Intelligence*, 97(1-2):45–82.

Dastani, M., Boer, F., Dignum, F., and Meyer, J. (2003). Programming agent deliberation. In *Proceedings of the Autonomous Agents and Multi Agent Systems Conference 2003*, pages 97–104.

de Giacomo, G., , Y. L., and Levesque, H. (2000). Congolog, a concurrent programming language based on the situation calculus. *Artificial Intelligence*, 121:109–169.

Delgrande, J., Schaub, T., and Jackson, W. (1994). Alternative approaches to default logic. *Artificial Intelligence*, 70:167–237.

D'Inverno, M. and Luck, M. (1998). Engineering agentspeak(l): A formal computational model. *Journal of Logic and Computation*, 8(3):233–260.

Ghose, A. K. and Goebel, R. G. (1998). Belief states as default theories: Studies in non-prioritized belief change. In *proceedings of the 13th European Conference on Artificial Intelligence (ECAI98)*, Brighton, UK.

Ghose, A. K., Hadjinian, P. O., Sattar, A., You, J., and Goebel, R. G. (1998). Iterated belief change. *Computational Intelligence*. Conditionally accepted for publication.

Giordano, L. and Martelli, A. (1994). On cumulative default reasoning. *Artificial Intelligence Journal*, 66:161–180.

Hindriks, K., de Boer, F., van der Hoek, W., and Meyer, J.-J. (1999). Agent programming in 3apl. In *Proceedings of the Autonomous Agents and Multi-Agent Systems Conference 1999*, pages 357–401.

Levesque, H., R., R., Lesperance, Y., F., L., and R., S. (1997). Golog: A logic programming language for dynamic domains. *Journal of Logic Programming*, 31:59–84.

MaynardReidII, P. and Shoham, Y. (1998). From belief revision to belief fusion. In *Proceedings of the Third Conference on Logic and the Foundations of Game and Decision Theory (LOFT3)*.

Meyer, T., Ghose, A., and Chopra, S. (2001). Non-prioritized ranked belief change. In *Proceedings of the Eighth Conference on Theoretical Aspects of Rationality and Knowledge (TARK2001)*, Italy.

Poole, D. (1988). A logical framework for default reasoning. *Artificial Intelligence*, 36:27–47.

Rao, A. S. (1996). Agentspeak(l): Bdi agents speak out in a logical computable language. *Agents Breaking Away, Lecture Notes in Artificial Intelligence*.

Rao, A. S. and Georgeff, M. P. (1991). Modeling rational agents within a bdi-architecture. In *Proceedings of the Second International Conference on Principles of Knowledge Repersentation and Reasoning (KR'91)*, pages 473–484.

Rao, A. S. and Georgeff, M. P. (1995). Bdi agents: From theory to practice. In *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95)*, San Francisco, USA.

Reiter, R. (1980). A logic for default reasoning. *Artificial Intelligence*, 13(1-2):81–132.

Riemsdijk, B., Hoek, W., and Meyer, J. (2003). Agent programming in dribble: from beliefs to goals using plans. In *Proceedings of the Autonomous Agents and Multi Agent Systems Conference 2003*, pages 393–400.

Shoham, Y. (1993). Agent-oriented programming. *Artificial Intelligence*, 60:51–93.

Wobcke, W. (2002). Intention and rationality for prs-like agents. In *Proceedings of the 15 Australian Joint Conference on Artificial Intelligence (AJAI02)*.

Wooldridge, M. (2000). *Reasoning about Rational Agent*. The MIT Press, London, England.