

Operational Semantic of an AgentSpeak(L) Interpreter Using Late Bindings

Frantisek Vidensky^a, Frantisek Zboril^b, Radek Koci^c and Frantisek V. Zboril^d

Department of Intelligent Systems, Brno University of Technology, Bozotechnova 2, Brno, Czech Republic

Keywords: BDI Agents, Operational Semantics, AgentSpeak(L).

Abstract: Although BDI systems have long been studied in the field of agent-based programming, there are still problems open for research. One problem is that some parts of systems are non-deterministic in the original specification. However, finding a suitable deterministic method can lead to improved rationality of an agent's behaviour. In our previous work, we introduced late binding into the interpretation of AgentSpeak(L) language. The main benefit of this approach is that the interpreter chooses substitutions only when needed, thus avoiding unnecessary and incorrect substitution selection. In this paper, we present a formal operational semantics for an interpreter using late binding variables. A well-specified operational semantics is necessary for the implementation of such an interpreter and its further development.

1 INTRODUCTION

Languages based on BDI paradigms have an important place in programming agents as autonomous proactive systems. Currently, one of the basic languages is AgentSpeak(L) (Rao, 1996) and its dialects, such as the ASL language for the Jason system (Bordini et al., 2007). In the paper (Rao, 1996), where the original language was first introduced, several selection functions were listed that fulfilled their role in agent control only in the abstract and introduced non-determinism into agent functionality. In the practical reasoning phase, these were functions for selecting a goal to follow and for choosing between suitable plans for the selected goal. In the execution phase, it was a selection function to choose an intention to be followed in a given cycle. From then to the present, a number of approaches have been published to address these non-determinisms to improve the rationality of the behaviour of such agents. For example, in (Caillou et al., 2017) the authors presented a cognitive agent architecture based on the BDI paradigm using priorities in plan selection. In (Nunes and Luck, 2014), softgoals were used to influence the selection of a plan to achieve goals. The authors of paper (Wa-


ters et al., 2015) introduced two new approaches for intention selection. The same team of authors in the paper (Waters et al., 2018) used a partial plan that does not specify the exact order of operations in the body of a plan, leading to increasing the agent's flexibility and robustness. Paper (Yao and Logan, 2016) introduced the use of the Monte Carlo Tree Search method to select intention to avoid conflicts in concurrently executing plans.


One additional problem that we believe has not received as much attention in this system has been the choice of substitutions during both phases, i.e., the reasoning and execution phases. In a similar BDI system, dMars (d'Inverno et al., 1997), the need to choose substitutions was also mentioned but also without detailed methods for choosing them. The more advanced CAN (Sardina and Padgham, 2011) system mentions substitutions as options for choosing plans, which are chosen if the previously chosen plan instance (i.e., the plan and any substitutions) fails.


In this paper, we want to demonstrate the operational semantics defining transition relation for interpreting the AgentSpeak(L) language using late bindings variables. We introduced this approach in the (Zboril Jr et al., 2022) paper.

This paper is structured as follows. The following section gives a very brief overview of AgentSpeak(L) and its constructs (such as beliefs, goals and plans). Sections 3 and 4 will describe the fundamental operations for late binding and its use for interpreting an

^a  <https://orcid.org/0000-0003-1808-441X>

^b  <https://orcid.org/0000-0001-7861-8220>

^c  <https://orcid.org/0000-0003-1313-6946>

^d  <https://orcid.org/0000-0002-6965-4104>

agent program. We then present the main contribution of this work, i.e. the operational semantics for interpreting the AgentSpeak(L) language using late binding variables in Section 5. Future work is discussed in the last section.

2 AGENT PROGRAM WRITTEN AgentSpeak(L)

This section will describe the basics of an agent program written in AgentSpeak(L). The following text is based on (Rao, 1996).

The essential element in an agent program is the agent, which is defined as follows:

An agent is given by a tuple $\langle PB, EQ, BB, AS, IS, S_E, S_O, S_I \rangle$, where PB is a plan base, EQ is an event queue, BB is a belief base, AS is a set of actions that are executable by the agent, IS is a set of intention structures and S_E, S_O, S_I are events, options and intentions selection functions.

The purpose of an agent is to achieve certain goals. There are two types of goals: achievement goals and test goals. Goals are written as $!g(\mathbf{t})$ and $?g(\mathbf{t})$ respectively, where g is a predicate symbol and \mathbf{t} is a sequence of terms. Achievement goals state that the agent wants to achieve an environmental state where the atom representing the goal is true. The test goal says that the agent wants to test whether there is unification with a belief from the base BB , which is a set of beliefs in the form of literal.

Essential elements of an agent program are plans that specify how to achieve a goal. Plans are written in the form of $t_e : \Psi \leftarrow plan$. Each plan consists of the triggering event t_e , context conditions Ψ and the body of the plan is composed of a sequence of elements that may be a goal or an action. A trigger event can be internal (when a sub-goal needs to be achieved) or external (triggered when the belief base is updated while observing the environment). Generally, an event is triggered by adding (prefix +) or removing (-) a belief or a goal.

For clarification, we mention that beliefs, events and goals are written as first-order logic atomic formulas (atoms). Also, a context condition for a plan is either atomic formula or a set of atomic formulas in conjunction.

In each cycle, an event is selected for processing using the function S_E . This event is then unified with the triggering events of the plans in the plan base PB . Plans whose triggering events are unified are called relevant plans. Relevant plans, whose context conditions are a logical consequence of the belief base, are called applicable plans. One of the applicable plans

is then selected using the function S_O . This plan is called a possible means of achieving the goal.

The problem that the system using late variables binding during interpretation solves is that the substitutions selected in one step may no longer be applicable in the next step. In that case, the whole deliberation cycle would have to be repeated, which can be a big problem in a very dynamic environment. Our proposed system uses delayed selection of substitution until necessary to solve this problem. This principle is called late bindings. The basic methods used in this system are described in the following section.

3 OVERVIEW OF OPERATIONS AND FUNCTIONS FOR LATE BINDINGS

As mentioned, in our previous article (Zboril Jr et al., 2022) we defined operations and functions for late bindings. For the purpose of completeness, they will be re-introduced and briefly described in this section.

Unifiers and substitutions play an essential role in late bindings, as they are created and then modified in each part of the agent's interpretation cycle. The purpose of the basic function is to create sets of unifiers. We called this function broad unification.

Definition 1. Broad unification, denoted as ρU , is formally defined as:

$$\rho U(p, PS) \stackrel{\text{def}}{=} \{mgu(p, p') : p' \in PS\}$$

maps the atom p and atom p' from the set of atoms PS to a set of all possible most general unifiers without variables renaming.

The result of a broad unification is a set of unifiers, which we call a possible unifier set (PUS). In the following text, we will use the simplified form ρU for the broad unification of the atom p and a set of atoms.

Definition 2. The instance set is denoted by $I\sigma$. This function maps an atom and a PUS to a set of atoms. This instance set is defined as follows:

$$I\sigma(p, \rho U) \stackrel{\text{def}}{=} \{p\sigma : \sigma \in \rho U\}$$

Informally, the instance set contains each atom that is created after applying each unifier from the PUS to an atom. It must be noted that the instance set is not the inverse function of the broad unification.

Definition 3. The Shorting function which we denoted \prec , is defined as follows:

$$\rho U \prec p \stackrel{\text{def}}{=} \{\sigma : \exists \sigma' (\sigma' \in \rho U, \sigma \subseteq \sigma', \forall [t/x] \in \sigma' (x \in \text{Var}(p) \rightarrow [t/x] \in \sigma))\}$$

The resulting set contains such substitutions that substitute free variables from p .

The following functions and operations are used to modify PUSs and play an essential role in late bindings. The first is the merging operation. Its purpose is to create a substitution from two other substitutions. The operation unites substitutions when each variable is substituted for the same term in both of them. If a situation arises where the substitutions map the same variable to two different terms, the result of the operation is an empty set.

Definition 4. The merging operation is denoted as \times and is defined as follows:

$$\sigma_1 \times \sigma_2 \stackrel{\text{def}}{=} \begin{cases} \sigma_1 \cup \sigma_2 & \text{iff } \forall [t_1/x_1] \in \sigma_1 \forall [t_2/x_2] \in \sigma_2 (x_1 = x_2 \rightarrow t_1 = t_2) \\ \emptyset & \text{else} \end{cases}$$

Assume that the merging substitutions unify two different atoms in two belief bases. If the result is a non-empty set, both substitutions unify both atoms into the belief bases. Nevertheless, there may be another pair of unifiers for which this operation produces a non-empty set. To find such a pair, we defined the restriction operator.

Definition 5. We denoted the restriction operator by \sqcap , this operator is defined as:

$$\rho U_1 \sqcap \rho U_2 \stackrel{\text{def}}{=} \bigcup_{\sigma_1 \in \rho U_1, \sigma_2 \in \rho U_2} \sigma_1 \times \sigma_2$$

The result of this operator for two PUS $\rho U(p_1, BB_1)$ and $\rho U(p_2, BB_2)$ is a set of unifiers that contain all the most general unifiers that unify p_1 in the belief base BB_1 and p_2 in the belief base BB_2 .

To transfer substitutions when moving from one plan to another, we introduced the PUS intersection function.

Definition 6. The intersection function denoted by \sim for PUS ρU and two atoms p_1 and p_2 is defined as follows:

$$p_1, \rho U \sim p_2 \stackrel{\text{def}}{=} \rho U(p_2, I\sigma(p_1, \rho U))$$

4 LATE BINDINGS IN INTERPRETATION OF AgentSpeak(L)

The reasoning process produces substitutions that are used in plan selection. It also creates substitutions by unifying a plan's triggering event and context conditions with the selected (by the S_e selection function) event to process and the agent's belief base.

In most BDI systems, substitution must be selected immediately when a plan is selected as a means to achieve a goal. Nevertheless, the late binding system will keep the substitutions separate as a PUS. We call this PUS context of the plan. This context changes when the agent performs an action or achieves a goal. The system can also assign context to events. Plans and events that do not have a selected substitution but have an associated context are called weak instances of plans and events.

Definition 7. A weak plan instance is a triple $\langle te, \mathbf{h}, ctx \rangle$, where te is a plan's triggering event, $\mathbf{h} = h_1; h_2; \dots; h_m$ is the plan's body, and ctx is the plan's context.

Similarly, we can define a weak event instance.

Definition 8. A weak event instance is a triple $\langle evt, ix, ctx \rangle$, where evt is an event, ix is an identifier of the intention that raises the event (or *null* in case of an external event) and ctx is a context.

We should note here that in the previous paper (Zboril Jr et al., 2022), the weak event instance was defined as a tuple, and the identifier of the intention was missing. This was for simplicity, as we did not need to distinguish between internal and external events. However, this will be necessary for this paper, as you will see later.

The purpose of a weak event instance is to represent an event that has arisen during the execution of a weak plan instance. Thus, the context of the currently executing plan is used to create this weak event instance.

In the following text, we will use the abbreviation WPI for weak plan instance and WEI for weak event instance.

The last definition in this section will be the definition of intention.

Definition 9. The intention is a structure containing WEIs of a plan's triggering event and a stack of WPI plans. Formally, it is defined as follows:

$$\langle evt, ix, ctx \rangle [\langle te_1, \mathbf{h}_1, ctx_1 \rangle \ddagger \langle te_2, \mathbf{h}_2, ctx_2 \rangle \ddagger \dots \ddagger \langle te_n, \mathbf{h}_n, ctx_n \rangle]$$

where the top of the stack is on the left.

If we need to shorten the intention notation, we replace part or all of the stack contents with \mathbf{P} . So $\langle evt, ix, ctx \rangle [\langle te_n, \mathbf{h}_n, ctx_n \rangle \ddagger \mathbf{P}]$ is also an intention.

In this and the previous sections the fundamentals of late binding have been presented, and in the next section the core of this paper, operational semantics, will be described.

5 OPERATIONAL SEMANTICS

A system using late binding variables performs a program written in the AgentSpeak(L) language and interprets it using weak instances. Its functioning can be described by certain transition rules that specify its operational semantic (Plotkin, 1981). This is a common instrument for precise formal specification of system behaviour based on labelled transition rules which define the steps in which a system may evolve. It has been used many times in the area of agent systems; for example, for AgentSpeak(L) interpretation in its basic version (Moreira and Bordini, 2002) as well as in the extended version with speech acts (Moreira et al., 2003), goal dynamic in CAN (Harland et al., 2014), etc.

Labelled transition rules define a relation among an agent's configurations.

Definition 10. Agent configuration is a tuple $\langle PB, EQ, BB, AS, IS \rangle$, where PB is a set of plans, EQ is an event queue that contains WEIs, BB is a belief base which consists of ground atoms, AS is a set of actions, and IS is a set of intentions.

When an agent is executed, it's running in a sequence of configurations from an initial configuration to a final one. The agent's current configuration changes either when it does some reasoning, or when it executes a plan. Let there be a relation \xrightarrow{AEX} (agent's execution) that is composed of two relations \xrightarrow{RSN} (reasoning) and \xrightarrow{ACT} (acting), where the first relation represents the change of the agent's configuration during its practical reasoning process and the second relation represents the agent's actions. Then we define that $\xrightarrow{AEX} = \left(\xrightarrow{RSN} \cup \xrightarrow{ACT} \right)$, which means that the AEX relation represents both practical reasoning as well as acting.

The transition rules for the agent's execution cover both these relations. Recall that the agent first looks for a means for a goal event. The means should be a plan which is relevant to the event, and it is applicable due to an agent's beliefs about the current environment. In this paper, we do not consider any advanced reasoning about event selection; we assume that an event is selected as the first element in the event queue if the queue is not empty. It differs from the original interpretation in the way it recognises applicability and relevance.

5.1 Relevant and Applicable Plans

For completeness, here is the definition of a relevant and applicable plan in the system. More about practi-

cal reasoning for weak instances can be found in our previous paper (Zboril Jr et al., 2022).

Assume that there is WEI and a plan p , then the plan p is relevant to the WEI when the PUS intersection of its triggering event and the event with context create a set containing at least one substitution. Note that this is true even if the substitution is an empty set. Formally:

Definition 11. A plan $te : b_1 \wedge b_2 \wedge \dots \wedge b_n \leftarrow h_1; h_2; \dots; h_m$ is relevant to a weak event instance $\langle evt, ix, ctx \rangle$ when $(evt, ctx \sim te) \neq \emptyset$

For a relevant plan to be applicable, all its context conditions must be satisfied in the belief base. More concretely, it must be possible to find a PUS for each context condition and belief base and then make the restriction between them.

Definition 12. A plan $te : b_1 \wedge b_2 \wedge \dots \wedge b_n \leftarrow h_1; h_2; \dots; h_m$ is applicable in the current belief base BB if it is true that:

$$\rho U(b_1, BB) \sqcap \dots \sqcap \rho U(b_n, BB) \neq \emptyset$$

One plan is then selected from the applicable plans using the selection function S_O .

Any plan that is relevant and applicable can be considered a means to achieve a goal, and the agent may choose it as its intended means. If the plan is both relevant and applicable, then the restriction of PUSs (contexts) for both WPI and WEI must be a non-empty PUS. Using the PUS intersection, we can write:

$$ctx_1 = ((evt, ctx \sim te) \sqcap \rho U(b_1, BB) \sqcap \dots \sqcap \rho U(b_n, BB)) \neq \emptyset$$

and then ctx_1 is a context of a plan which, together with the plan's body $h_1; h_2; \dots; h_m$, forms a new WPI that can be an intended means for the WEI.

5.2 Transition Rules

In the following text, we can divide the presented rules into three groups to define the relations mentioned above. The first group defines the relation \xrightarrow{RSN} using two rules.

The first rule ($PR - EXTEVT$) is used in situations when the agent deals with an external event and the second rule ($PR - INTEVT$) is for the case of an internal event. Both these rules change the agent's event queue and intention structure. Both also compute contexts for relevant and applicable plans, which can all be an intended means for the event. Whether the event is internal or external depends on the form

$$\begin{array}{c}
\text{PR-EXTEVT} \frac{
\begin{array}{l}
e = \langle \text{evt}, \text{null}, \text{ctx} \rangle \in EQ \\
p = te : b_1 \wedge \dots \wedge b_n \leftarrow \mathbf{h} \in PB \\
\text{ctx}_1 = ((\text{evt}, \text{ctx} \sim te), \rho U(b_1, BB) \sqcap \dots \sqcap \rho U(b_n, BB)) \neq \emptyset
\end{array}
}{
\langle PB, EQ, BB, AS, IS \rangle \xRightarrow{RSN} \langle PB, EQ - \{e\}, BB, AS, IS \cup \{ \langle \text{evt}, ix, \text{ctx} \rangle [\langle te, \mathbf{h}, \text{ctx}_1 \rangle] \} \rangle
} \\
\\
\text{PR-INTEVT} \frac{
\begin{array}{l}
e = \langle \text{evt}, ix, \text{ctx} \rangle \in EQ \\
p = te : b_1 \wedge \dots \wedge b_n \leftarrow \mathbf{h} \in PB \\
\text{ctx}_1 = ((\text{evt}, \text{ctx} \sim te), \rho U(b_1, BB) \sqcap \dots \sqcap \rho U(b_n, BB)) \neq \emptyset
\end{array}
}{
\langle PB, EQ, BB, AS, IS \rangle \xRightarrow{RSN} \\
\langle PB, EQ - \{e\}, BB, AS, (IS - \{ \langle \text{evt}_2, ix, \text{ctx}_2 \rangle [\mathbf{P}] \}) \cup \{ \langle \text{evt}_2, ix, \text{ctx}_2 \rangle [\langle te, \mathbf{h}, \text{ctx}_1 \rangle \ddagger \mathbf{P}] \} \rangle
} \\
\\
\text{EXEC1} \frac{
\begin{array}{l}
i = \langle \text{evt}, ix, \text{ctx} \rangle [\langle te_1, \mathbf{h}_1, \text{ctx}_1 \rangle] \in IS_1 \\
\langle EQ_1, BB_1, AS_1, (ix : \mathbf{h}_1, \text{ctx}_1) \rangle \rightarrow \langle EQ_2, BB_2, AS_2, (ix : \mathbf{h}_2, \text{ctx}_2) \rangle
\end{array}
}{
\langle PB_1, EQ_1, BB_1, AS_1, IS_1 \rangle \xRightarrow{ACT} \\
\langle PB_1, EQ_2, BB_2, AS_2, (IS_1 - \{i\}) \cup \{ \langle \text{evt}, ix, \text{ctx} \rangle [\langle te_1, \mathbf{h}_2, \text{ctx}_2 \rangle] \} \rangle
} \\
\\
\text{EXEC2} \frac{
\begin{array}{l}
i = \langle \text{evt}, ix, \text{ctx} \rangle [\langle te_1, \mathbf{h}_1, \text{ctx}_1 \rangle \ddagger \mathbf{P}] \in IS_1 \\
\langle EQ_1, BB_1, AS_1, (ix : \mathbf{h}_1, \text{ctx}_1) \rangle \rightarrow \langle EQ_2, BB_2, AS_2, (ix : \mathbf{h}_2, \text{ctx}_2) \rangle
\end{array}
}{
\langle PB_1, EQ_1, BB_1, AS_1, IS_1 \rangle \xRightarrow{ACT} \\
\langle PB_1, EQ_2, BB_2, AS_2, (IS_1 - \{i\}) \cup \{ \langle \text{evt}, ix, \text{ctx} \rangle [\langle te_1, \mathbf{h}_2, \text{ctx}_2 \rangle \ddagger \mathbf{P}] \} \rangle
} \\
\\
\text{CLEARINT1} \frac{
i = \langle \text{evt}, ix, \text{ctx} \rangle [\langle te_1, \text{null}, \text{ctx}_1 \rangle] \in IS
}{
\langle PB, EQ, BB, AS, IS \rangle \xRightarrow{ACT} \langle PB, EQ, BB, AS, (IS - \{i\}) \rangle
} \\
\\
\text{CLEARFAIL1} \frac{
i = \langle \text{evt}, ix, \text{ctx} \rangle [\langle te_1, \text{fail}, \text{ctx}_1 \rangle] \in IS
}{
\langle PB, EQ, BB, AS, IS \rangle \xRightarrow{ACT} \langle PB, EQ \cup \{ \langle \text{evt}, \text{null}, \text{ctx} \rangle \}, BB, AS, (IS - \{i\}) \rangle
}
\end{array}$$

of the WEI. When there is no parent intention mentioned, better say it is null, then a new intention stack is created and an intended means is inserted there. Otherwise, the means are added to the corresponding intention stack.

If all three conditions above the line are satisfied (*null* in this rule means that no intention is bound to the event), then WEI e is removed from an event queue EQ and a new intention ix is created. This intention is composed of new WEIs $\langle \text{evt}, ix, \text{ctx} \rangle$ and a plan stack that contains WPI $\langle te, \mathbf{h}, \text{ctx}_1 \rangle$. Note that te is a triggering event of the chosen plan, \mathbf{h} is its body, and ctx_1 is the context computed as was shown in Definitions 11 and 12.

The $PR - INTEVT$ rule is similar to the previous one. It differs only in the form of the processed WEI.

The second group of rules defines the \xRightarrow{ACT} relation. At the intention level, the agents try to execute one step for an intention from its intention set (the intention was created for some WEI $\langle \text{evt}_2, ix, \text{ctx}_2 \rangle$). There are six rules ($EXEC1$, $EXEC2$, $CLEARINT1$, $CLEARFAIL1$, $CLEARINT2$, $CLEARFAIL2$) that de-

fine how the intention can change when the agent performs the first item of its top-level plan. These rules are used in situations when an intention step finishes either successfully or unsuccessfully. If the step was successful, then there are two separate rules for when it completes the entire plan, and for when any actions remain in the body of the plan. We must distinguish whether the completed plan was a top-level plan of the intention or a sub-plan within the intention.

All these rules use one more lower-level transition relation that determines the behaviour of the agent at the plan level. This transition relation is denoted as \rightarrow and represents one step in the plan execution. The following rules transform n-tuples of the form $\langle EQ, BB, AS, (ix : \mathbf{h}, \text{ctx}) \rangle$, where EQ , BB and AS are the same as in Definition 10. Sets PB and IS are omitted and instead there is a shortened version of an intention stack that contains its identifier ix , \mathbf{h} is the body of its top-level plan, and ctx is its context.

The $EXEC1$ and $EXEC2$ rules are defined for any non-empty intention stack with a non-empty plan on top of it. Furthermore, the execution of a plan is as-

sumed to lead neither to an empty plan nor to its failure. The body of the plan should remain in the intention stack, but its body is always truncated by one item. In addition, other parts within the configuration may also be changed.

The first rule is applicable if the WIP is the only element of the intention stack, and the second rule is applicable if there is more than one WIP in the intention stack.

When the agent completes a sub-plan, the sub-plan is removed from the intention stack. A sub-plan is completed when there are no more items in the body of the plan, which is represented by null. Nevertheless, if the plan is the last plan in the intention stack, then the agent successfully achieves the top-level goal and the corresponding plan is also removed from the *IS*. Failure to complete the plan will result in the removal of the corresponding plan, and moreover, the WEI must be put back into the *EQ*.

The situation is a little more complicated when the agent completes a plan to achieve a goal. Some information should be transmitted upward to the goal-setting plan. The just-completed plan contains a context with substitutions corresponding to all the goals the agent has achieved during execution.

The higher-level plan declared the achievement goal $!g(\mathbf{t}_1)$ in the current context ctx_1 . For this goal, a plan with trigger event $+!g(\mathbf{t}_2)$ was chosen. Assume that this plan has been successfully completed with a context ctx_2 . If we create an instance set from the trigger event $g(\mathbf{t}_2)$ of the plan and the context ctx_2 , we get all the goals that the plan achieved. Using the intersection $g(\mathbf{t}_2), ctx_2 \sim g(\mathbf{t}_1), ctx_1$ of the achievement

goal $g(\mathbf{t}_1)$ in the context ctx_1 with the triggering event $g(\mathbf{t}_2)$ in the context ctx_2 we obtain a new context ctx_3 .

Note that the body of the plan in the first line consists of two parts - the abstract part denoted as \mathbf{h}_1 and then the achievement goal $!g(\mathbf{t}_1)$. Only the first part of \mathbf{h}_1 remains after this step.

If the plan fails and is not the last plan in the intention stack, then the plan is removed from the stack and the higher-level plan still contains the achievement goal.

The third and final set of rules defines the agent's behaviour at the plan level.

In the late binding system, the test target is the only plan item that can fail. Whether it succeeds depends on the result of the restriction of the goal atom, the current context of the plan, and the current state of the agent's belief base. If the result is an empty set, it means that the plan can continue to execute, and the set is the new context of the plan.

In case the test goal fails, the whole plan fails.

An achievement goal may cause a new weak event instance. If there is no corresponding WEI in the agent's *EQ* and the intention stack for the WEI does not yet exist, the agent creates a new weak event instance and puts it into the agent's *EQ* event queue. Otherwise, the achievement goal is already processed or ready to be processed so the agent ignores it. Thus, if a new WEI is created, then it consists of the achievement goal atom, an intention identifier, and a context that is computed from the context of the original plan truncated to the variables of the achievement goal. In this case, the achievement goal is not removed from the plan and remains in the body of the plan until the agent achieves it.

$$\begin{array}{l}
 \text{CLEARINT2} \frac{i = \langle evt, ix, ctx \rangle [\langle te_1, !g(\mathbf{t}_1); \mathbf{h}_1, ctx_1 \rangle \ddagger \langle +!g(\mathbf{t}_2), null, ctx_2 \rangle \ddagger \mathbf{P}] \in IS}{\langle PB, EQ, BB, AS, IS \rangle \xrightarrow[ACT]{} \langle PB, EQ, BB, AS, (IS - \{i\}) \cup \{ \langle evt, ix, ctx \rangle [\langle te_1, \mathbf{h}_1, ctx_3 \rangle \ddagger \mathbf{P}] \} \rangle} \\
 \text{CLEARFAIL2} \frac{i = \langle evt, ix, ctx \rangle [\langle te_1, !g(\mathbf{t}_1); \mathbf{h}_1, ctx_1 \rangle \ddagger \langle te_2, fail, ctx_2 \rangle \ddagger \mathbf{P}] \in IS}{\langle PB, EQ, BB, AS, IS \rangle \xrightarrow[ACT]{} \langle PB, EQ, BB, AS, (IS - \{i\}) \cup \{ \langle evt, ix, ctx \rangle [\langle te_1, !g(\mathbf{t}_1); \mathbf{h}_1, ctx_1 \rangle \ddagger \mathbf{P}] \} \rangle} \\
 \text{TESTG} \frac{ctx_1 = ctx \sqcap \rho U(g(\mathbf{t}), BB) \neq \emptyset}{\langle EQ, BB, AS, (ix : ?g(\mathbf{t}); \mathbf{h}, ctx) \rangle \rightarrow \langle EQ, BB, AS, (ix : \mathbf{h}, ctx_1) \rangle} \\
 \text{TESTFL} \frac{ctx \sqcap \rho U(g(\mathbf{t}), BB) = \emptyset}{\langle EQ, BB, AS, (ix : ?g(\mathbf{t}); \mathbf{h}, ctx) \rangle \rightarrow \langle EQ, BB, AS, (ix : fail, ctx) \rangle} \\
 \text{ACHIEVEG} \frac{ctx_1 = ctx \prec q(\mathbf{t})}{\langle EQ, BB, AS, (ix : !q(\mathbf{t}); \mathbf{h}, ctx) \rangle \rightarrow \langle EQ \cup \{ \langle !q(\mathbf{t}), ix, ctx_1 \rangle \}, BB, AS, (ix : !q(\mathbf{t}); \mathbf{h}, ctx) \rangle}
 \end{array}$$

$$\begin{array}{c}
\text{ADDBB} \frac{\sigma \in (ctx \prec c(\mathbf{t}))}{\langle EQ, BB, AS, (ix : +c(\mathbf{t}); \mathbf{h}, ctx) \rangle \rightarrow \langle EQ \cup \{+c(\mathbf{t})\sigma, ix, ctx\}, BB \cup \{c(\mathbf{t})\sigma\}, AS, (ix : \mathbf{h}, ctx \sqcap \{\sigma\}) \rangle} \\
\text{DELBB} \frac{\sigma \in (ctx \prec c(\mathbf{t}))}{\langle EQ, BB, AS, (ix : -c(\mathbf{t}); \mathbf{h}, ctx) \rangle \rightarrow \langle EQ \cup \{-c(\mathbf{t})\sigma, ix, ctx\}, BB \setminus \{c(\mathbf{t})\sigma\}, AS, (ix : \mathbf{h}, ctx \sqcap \{\sigma\}) \rangle} \\
\text{EXTACT} \frac{\sigma \in (ctx \prec c(\mathbf{t}))}{\langle EQ, BB, AS, (ix : c(\mathbf{t}); \mathbf{h}, ctx) \rangle \rightarrow \langle EQ, BB, AS \cup \{c(\mathbf{t})\sigma\}, (ix : \mathbf{h}, ctx \sqcap \{\sigma\}) \rangle}
\end{array}$$

The previous rules changed the context using restrictions and other functions and operations introduced in section 3. External actions, as well as both types of internal actions, must be precisely specified. In other words, the agent must know exactly what to do. For this reason, the action atom must be ground and the agent must decide which concrete substitution or substitutions to use from the context of the plan. This does not mean that only one set of substitutions can remain in the context. Note that there can be more than one set of substitutions that map free variables of action atoms to the same terms. Again, the shorting function is used here, which takes a context with an action atom and maps them to one particular set of substitutions of that atom. If we use these substitutions to restrict the context of the original plan, we get a new context that suits the executed action.

The following three rules *ADDBB*, *DELBB* and *EXTACT* respond to the execution of an action during the execution of a plan. It may be the addition or removal of a belief from the belief base. Rules *ADDBB* and *DELBB* are used in these situations.

Similarly, we can define the *DELBB* rule. In both rules, the most suitable substitution is selected first and the action is removed from the body of the currently executing plan. A new WEI is then put into the agent's event queue *EQ*. The result of the restriction of the context of the plan and $\{\sigma\}$ is used as the new context for the top-level plan. The substitution σ is applied to the belief $c(\mathbf{t})$ and the result is inserted or removed from the belief base *BB*.

The last rule is used when an external action is performed.

Also in this rule, the substitution σ is chosen and is used to compute the new context. The chosen substitution is also applied to the external event $c(\mathbf{t})$ and the result is inserted into the set of action *AS*.

6 CONCLUSIONS

We consider the transition system introduced in this paper to be a basic formal specification of the interpretation of AgentSpeak(L) language using late bindings. Such an interpretation introduces more flexibility into agent decision-making by preserving options until a specific decision needs to be made to perform an action, compared to the original approaches. This agent does not give up options for its future actions prematurely and can correct its behaviour during the execution of each plan according to the current situation or its current beliefs about the state of the system. Currently, such an interpretation is implemented in the system we program in the PROLOG language. In this system, it can also be practically verified that for some agent programs in that language, the agent can deal with situations that fail in systems with classical interpretation. However, there are still a few open areas, for example, the intention selection function is not defined. Its implementation can be done in several ways, so it will be necessary to design several solutions and compare them with each other on a pre-prepared set of examples using well-defined metrics. This will be the direction of our future work.

ACKNOWLEDGEMENTS

This work has been supported by the internal BUT project FIT-S-20-6427.

REFERENCES

- Bordini, R. H., Hübner, J. F., and Wooldridge, M. (2007). *Programming multi-agent systems in AgentSpeak using Jason*, volume 8. John Wiley & Sons.
- Caillou, P., Gaudou, B., Grignard, A., Truong, C. Q., and Taillandier, P. (2017). A simple-to-use bdi architecture for agent-based modeling and simulation. In *Advances in Social Simulation 2015*, pages 15–28. Springer.

- d’Inverno, M., Kinny, D., Luck, M., and Wooldridge, M. (1997). A formal specification of dmars. In *International Workshop on Agent Theories, Architectures, and Languages*, pages 155–176. Springer.
- Harland, J., Morley, D. N., Thangarajah, J., and Yorke-Smith, N. (2014). An operational semantics for the goal life-cycle in bdi agents. *Autonomous agents and multi-agent systems*, 28(4):682–719.
- Moreira, Á. F. and Bordini, R. H. (2002). An operational semantics for a bdi agent-oriented programming language. In *Proceedings of the workshop on logics for agent-based systems (LABS-02), held in conjunction with the Eighth International Conference on Principles of Knowledge Representation and Reasoning (KR2002), April*, volume 22, pages 45–59.
- Moreira, Á. F., Vieira, R., and Bordini, R. H. (2003). Extending the operational semantics of a bdi agent-oriented programming language for introducing speech-act based communication. In *International Workshop on Declarative Agent Languages and Technologies*, pages 135–154. Springer.
- Nunes, I. and Luck, M. (2014). Softgoal-based plan selection in model-driven bdi agents. In *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*, pages 749–756.
- Plotkin, G. D. (1981). *A structural approach to operational semantics*. Aarhus university.
- Rao, A. S. (1996). Agentspeak (I): Bdi agents speak out in a logical computable language. In *European workshop on modelling autonomous agents in a multi-agent world*, pages 42–55. Springer.
- Sardina, S. and Padgham, L. (2011). A bdi agent programming language with failure handling, declarative goals, and planning. *Autonomous Agents and Multi-Agent Systems*, 23(1):18–70.
- Waters, M., Nebel, B., Padgham, L., and Sardina, S. (2018). Plan relaxation via action debinding and deordering. In *Twenty-Eighth International Conference on Automated Planning and Scheduling*.
- Waters, M., Padgham, L., and Sardina, S. (2015). Improving domain-independent intention selection in bdi systems. *Autonomous Agents and Multi-Agent Systems*, 29(4):683–717.
- Yao, Y. and Logan, B. (2016). Action-level intention selection for bdi agents. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems, AAMAS ’16*, page 1227–1236. International Foundation for Autonomous Agents and Multiagent Systems.
- Zboril Jr, F., Vidensky, F., Koci, R., and Zboril, F. V. (2022). Late bindings in agentspeak (I). In *ICAART (3)*, pages 715–724.