

# Late Bindings in AgentSpeak(L)

Frantisek Zboril<sup>1</sup><sup>a</sup>, Frantisek Vidensky<sup>1</sup><sup>b</sup>, Radek Koci<sup>1</sup><sup>c</sup> and Frantisek V. Zboril<sup>1</sup><sup>d</sup>

<sup>1</sup>*Department of Intelligent Systems, Brno University of Technology, Bozotechnova 2, Brno, Czech Republic  
{zborilf, ividensky, koci, zboril}@fit.vutbr.cz*

**Keywords:** BDI Agents, Planning, Agent Interpretation, AgentSpeak(L)

**Abstract:** For agents based on BDI theory, some problems remain open. These include parts of the interpretation of these systems that are nondeterministic in the original specifications, and finding methods for their determinism should lead to improved rationality of agent behaviour. These problems include the choice of a plan suitable for achieving the goal, then the choice of the intention to be pursued by the agent at any given time, and if a language based on predicate logic is used to implement such an agent, then there is also the problem of choosing variable substitutions. One such agent-based system is systems using the AgentSpeak(L) language, which will be the basis for this paper. We will introduce late binding into the interpretation of this language and show that they do not make the agent lose the possibility of achieving the goal by making unnecessary or incorrect substitutions in cases where such a decision is not necessary. We show that with late binding substitutions the agent operates with all possible substitutions given by the chosen plan to the goals in the plan structure, and that these substitutions are always valid with respect to the acts performed so far within this plan.


## 1. INTRODUCTION


One way to create artificial agents is based on Bratman's theory of intentions (Bratman, Intention, Plans and Practical Reason, 1987). Intentions, as the mental states of agents, are persistent goals (Cohen & Levesque, 1990), which an agent decides to achieve at the moment some of its desires have been found attainable and are pursued until they are achieved or the agent finds them unachievable. Systems combining intentions with other mental states, namely beliefs and desires, known as BDI systems (Rao & Georgeff, Modeling Rational Agents within a BDI-Architecture, 1991), have gained popularity. Implementations of such systems include IRMA (Bratman, Israel, & Pollack, Plans and resource-bounded practical reasoning, 1988), PRS (Georgeff & Lansky, 1987), dMars (d'Iverno, Kinny, Luck, & Wooldridge, 1998), 2APL (Dastani, 2008), CAN (Sardina & Padgham, 2011), as well as systems interpreting the AgentSpeak(L) language (Rao, AgentSpeak(L): BDI agents speak out in a logical computable language, 1996). The principle of these


systems is based on the choice of intentions from the desirec that the agent has, which may become intentions at an opportune moment based on the current state of the agent's beliefs. The plan then tries to fulfill this intention by executing a plan, or plans, that are hierarchically ordered. Each plan may declare goals to be achieved, and a subplan is chosen to achieve those goals. If the top-level plan that has been chosen to achieve the goal of an intention is achieved, that intention is also achieved.


The problems that are still a matter of research are found in the non-deterministic parts of an agent's decision making. This involves both the selection of the appropriate means to achieve goals or subgoals in the form of plans, but it also involves, when a language based on predicate logic is used, the choice of appropriate substitutions. In addition, the agent, if pursuing multiple goals simultaneously, also has the dilemma of which goal to pursue at any given step.

This text offers a solution for dealing with substitutions such that if a plan is chosen, then substitutions need not be applied immediately, but can be deferred until a decision needs to be made, typically before a particular action is taken. Poorly

<sup>a</sup> <https://orcid.org/0000-0001-7861-8220>

<sup>b</sup> <https://orcid.org/0000-0003-1808-441X>

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

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

chosen substitutions during the practical reasoning phase may indicate the choice of the wrong resource, which may become apparent later during the execution of the plan thus instantiated. Several ways of handling substitutions during agent execution are currently proposed. The 2APL system (Dastani, 2008) offers the possibility to create rules that specify decision processes including the handling of possible substitutions. For the CAN system (Sardina & Padgham, 2011), knowledge of the substitutions used is stored and, if a new plan has to be searched if a previous attempt to achieve the intentions failed with some substitutions, other possible substitutions can be chosen for the same plan.

Our approach is based on trying to preserve all possible variable bindings and during the execution of a plan, it works with all of them. If some step (act) of the plan is executed, then the number of these options may be limited and only some of them survive on. However, if at least one of the options persists, then the plan can continue. We first introduced the principle that leads to late variable binding in (Zboril, Koci, Janousek, & Mazal, 2008) and later discussed it in (Zboril, Zboril, & Kral, Flexible Plan Handling using Extended Environment, 2013). In this paper, we give a more precise and modified form of the basic operations we need to create a late-binding AgentSpeak(L) interpreter and specify the use of each operation here.

The individual sections are organized as follows. In Section 2, we briefly introduce the AgentSpeak(L) language and give the motivation for introducing late binding for its interpreter. Section 3 defines the operations that are necessary for such an interpretation, and in Section 4, we introduce the concept of weak plans and events, which we build on by introducing the execution of a single plan or a hierarchy of plans with a late binding approach.

## 2. EXECUTION OF AGENTSPEAK(L)

An agent that is executed according to some program written in AgentSpeak(L), is 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 functions for selections of events, options and intentions (Rao, AgentSpeak(L): BDI agents speak out in a logical computable language, 1996). The program itself written in this language defines knowledge as formulas of first-order predicate logic, for simplicity we will consider beliefs in the form of atomic

formulas of this logic as facts are defined in PROLOG as a predicate. Goals can be declared as achievement or test goals. In the first case, the predicate is given with the preposition ! and in the latter with the preposition ?. The core part of a program in AgentSpeak(L) is the plan of how to achieve some goal in the form of plans. This looks like the following  $t_e: \Psi \leftarrow plan$  and consists of the triggering event  $t_e$ , context conditions  $\Psi$  and plan's body. Triggering events are written in the form +event or -event, where an event can be a test goal, an achievement or just a predicate. A plan is then relevant to some goal in the form of an event, when its triggering event is unifiable with that event, and it is further applicable if the context conditions are valid in the current state of the agent's BB. If we have a substitution for which the plan is relevant and applicable, then the plan is a possible means of achieving the goal.

The interpretation of an agent programmed in AgentSpeak(L) is presented in the original paper (Rao, AgentSpeak(L): BDI agents speak out in a logical computable language, 1996), and the operational semantics is presented more formally in (Winikoff, 2005). The problem we address and offer a solution to is that if a plan is chosen, a substitution is chosen that may not be appropriate for the agent's next acts. As difficult as it is to predict which of the possible behaviors will lead to success in a dynamic environment, it is possible to either try to estimate these substitutions based on previous experience, or to defer the decision about substitutions until this becomes necessary. We use the latter approach and now wish to demonstrate its potential appropriateness and usefulness.

Let us return to the aforementioned interpretation of plan selection for some goal in the form of an event  $e$ . Then there may exist one or more plans from agent's plan base such that their triggering events are unifiable with  $e$  for some most general unifier (mgu)  $\sigma$ . Thus, let us have such plans  $te_1: \Psi_1 \leftarrow plan_1$ ,  $te_2: \Psi_2 \leftarrow plan_2$  ...  $te_n: \Psi_n \leftarrow plan_n$  and  $\sigma_1 = mgu(e, te_1)$ ,  $\sigma_2 = mgu(e, te_2)$  ...  $\sigma_n = mgu(e, te_n)$  are the mgu for the event and individual triggering events. These plans are relevant but only applicable if  $BB| = \Psi_1 \sigma_1$  for the first plan, etc. Thus, again, we look for substitutions that unify the individual context conditions in the agent's BB, let's denote them as  $\rho_1, \rho_2 \dots \rho_n$ . Then the agent selects one of these plans to achieve the goal represented by event  $e$ , say plan  $j$ , and the agent is going to execute the body of this plan with the appropriate substitutions to achieve the event.

Thus, only one option is chosen from all that can be found. Not only is it potentially possible to find multiple plans for a single goal, but also it is possible

to find multiple substitutions for which these plans are relevant and applicable.

By introducing the context as a set of possible substitutions, we generalize the execution of one plan to the execution of several possibilities found for that plan during practical reasoning. We will refer to a plan as a structure given in the program, supplemented with context as a set of substitutions, as a weak instance of the plan. That is, if for the plan  $te_i: \Psi_i \leftarrow plan_i$  it is possible to find multiple substitutions that unify this plan's  $te_i$  with the event  $e_i$  and for which this plan is applicable in the agent's BB, let's say  $ctx$ , then  $\langle e_i: \Psi_i \leftarrow plan_i, ctx \rangle$  is a weak instance of the plan  $e_i: \Psi_i \leftarrow plan_i$ . The context  $ctx$  will allow to postpone the decision about substitutions until the moment when this is necessary. We therefore view this approach as a late binding for the interpretation of AgentSpeak(L). The weak plan instance is executed using the operations we present in the next section. After introducing them, we will also show how the acts in the weak plan instance are interpreted and how subplans and transitions between plan levels are handled, which will allow us to conclude by showing that such an interpretation is done correctly and that it expands the decision-making options for these types of agents.

### 3. OPERATIONS AND FUNCTIONS FOR LATE BINDINGS

The basic terms of the formal description of our approach are the sets of unifiers and substitutions. (Substitution is a set of tuples, in this text  $[t/x]$ , where  $x$  is variable and  $t$  is term. Unifier is a substitution that unifies some two formulas). Sets of substitutions play an essential role in our interpretation, as they are created and modified during agent's interpretation. For a specification on how the corresponding operations work, we must write some definitions. First, we introduce broad unification as a function that creates a set of unifiers.

**Definition 1:** Broad unification is denoted as  $\rho U$  is a function that maps the predicate  $p$  and predicates  $p'$  from the BB to a set of all possible mgu without variables renaming. Using function  $mgu(p, p')$  for finding the mgu of  $p$  and  $p'$  without variable renaming we define it formally as

$$\rho U(p, BB) \stackrel{\text{def}}{=} \{mgu(p, p'): p' \in BB\} \quad (1)$$

In this definition we say that BB is a set of predicates. Because beliefs are also in the form of predicates, then any BB is also a set of predicates.

Variable renaming is important in the unification for making the formulas' logical equivalent. However, in our case we need the unification just for specification of the predicate  $p$  in such a way that it is valid in the BB in any interpretation. For this reason, we need not unify the name of the variables. However, even though such a function from Definition 1 does not unify the predicates in the correct sense, we will use the /term unification for this anyway.

The result of the broad unification is a set of unifiers that we call to be a possible unifier set (in this text we write PUS). The set of predicates consists in this case of all beliefs that form the agent's BB. Further, in the text we will use a symbolic representation of the broad unification of a predicate  $p$  and a belief base BB in the simplified form  $\rho U_{BB}^p$ , or when the parts are not necessary, we will write just  $\rho U$ . Second, we use in some further definitions when we refer to the set just as certain PUSs and we do not need a precise determination of the predicate and belief base.

Conversely, an instance set is a set of predicates containing every predicate that rises, after the application of every unifier, from a PUS to a predicate.

**Definition 2:** We denoted the instance set as  $I\sigma$ , and it is a function that maps a predicate and a PUS to a set of predicates. We define it in the following way:  $I\sigma(p, \rho U) \stackrel{\text{def}}{=} \{p\sigma: \sigma \in \rho U\}$

Notice that an instance set is not inverse to the broad unification in the sense that if we have  $\rho U_{BB}^p$ , then  $I\sigma(p, \rho U_{BB}^p)$  need not be equal to the BB and vice versa.

**Definition 3:** Shorting is a function denoted by  $\prec$  and we define it as:

$$\rho U \prec p \stackrel{\text{def}}{=} \quad (2)$$

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

The resulting substitution set contains just those substitutions that substitute free variables from  $p$ . The  $\text{Var}$  function is used here in the usual way, which means that by using  $\text{Var}(p)$  we get a set of free variables in  $p$ . Then, only the variable substitutions from  $\rho U$  unifiers that substitute any of the  $\text{Var}(p)$  persist in  $\rho U \prec p$ .

The functions and operations that we call merging, restriction, shorting, and decision-making are used for the transformation of PUSs and play an essential role for late bindings. The purpose of the first operation, which we call 'merging', is to create

substitution from two other substitutions. It unites the substitutions when every variable substituted in both of them is substituted for the same term. This means that there must not be a conflict where the substitutions map the same variable to two different terms. If this occurs, then the result of the merging is an empty set.

**Definition 4:** We denote the merging by  $\bowtie$  and it is defined for a certain two unifiers as follows:

$$\begin{aligned} \sigma_1 \bowtie \sigma_2 &\stackrel{\text{def}}{=} & (3) \\ &\sigma_1 \cup \sigma_2 \text{ iff } \forall [x_1/t] \in \sigma_1 \forall [x_2/t] \in \sigma_2 (x_1 = x_2 \rightarrow t_1 = t_2) \\ &\{ \} \text{ else} \end{aligned}$$

Let the first substitution be a unifier for a predicate and a belief base, and the second unifies another predicate in another belief base. If the result of the merging of these unifiers is a nonempty set of substitutions, then this set of substitutions unify both predicates with some belief in the belief bases. If the result is an empty set, then the unifiers cannot be united. However, there can be another pair of unifiers for which the merging produces a non-empty set. For finding each such pair, we define the restriction operator.

**Definition 5:** The restriction operator is denoted as  $\sqcap$  and is defined for some two PUS  $\rho U_1$  and  $\rho U_2$  as

$$\rho U_1 \sqcap \rho U_2 \stackrel{\text{def}}{=} \cup_{\sigma_1 \in \rho U_1, \sigma_2 \in \rho U_2} \sigma_1 \bowtie \sigma_2 \quad (4)$$

Using the restriction, an agent obtains pairs of unifiers from both PUSs that work as extended unifiers for both pairs of predicate/belief bases for which it makes the original PUSs.

The following Theorem 1 shows a key property of this operation, which is fundamental for the functionality of the late bindings.

**Theorem 1:** The restriction of two PUSs created by some broad unifications  $\rho U_{BB_1}^{p_1}$  and  $\rho U_{BB_2}^{p_2}$  results in a set of unifiers that contains all the mgu that unify both  $p_1$  in belief base  $BB_1$  and  $p_2$  in another belief base  $BB_2$ .

**Proof:** First, we prove that all such unifiers work properly for both pairs of predicates and belief bases (1). Moreover, we need to show that they are in their most general form (2). Then we prove that there is no other mgu (3).

Ad 1) Let  $\sigma \in \rho U_{BB_1}^{p_1} \sqcap \rho U_{BB_2}^{p_2}$  be a unifier that unifies  $p_1$  in  $BB_1$  and  $p_2$  in  $BB_2$ . From Definitions 4 and 5 it follows that  $\sigma$  is a union of the original unifiers. If the predicates are unified to a ground

belief, then every variable in  $p_1$  and  $p_2$  must be substituted in the same way as they were substituted in the original unifiers. There cannot be a conflict, because if both unifiers substitute the same variable, by definition of the merging operator, it follows that it was substituted in the same way. Even if it unifies them with a lighted predicate, then one unifier may cause the substitution of a variable that remained free in the second unification; this means that, for example,  $\sigma$  substitutes a variable in  $p_1$ , but in the original  $\rho U_{BB_1}^{p_1}$  this variable remained free. However, this is just the specification in the predicate logic and by the rule of specification this substitution makes  $p_1$  also valid in  $BB$ .

ad 2) If  $\sigma$  is not the mgu for both unifications, then there must be a unifier  $\sigma_1$  and a substitution  $\delta$  where  $\sigma = \sigma_1 \delta$ . Definition 5 says that every mapping of a variable in  $\delta$  must also be in either  $\rho U_{BB_1}^{p_1}$  or  $\rho U_{BB_2}^{p_2}$ . If  $\sigma'$  works for both unifications, then there must be a unifier  $\sigma_2$  for the first or second unification which is more general, then a unifier from  $U_{BB_1}^{p_1}$  and  $\rho U_{BB_2}^{p_2}$ , respectively. However, because these unifiers are mgu, this is not possible. From another point of view, because unifications produce ground predicates, then the number of variable substitutions in the unifier must be equal to the number of free variables in  $p_1$  and  $p_2$  that is valid for the results of the restriction operation.

ad 3) Let us consider that there is, a unifier,  $\sigma$  that is mgu for both predicates and belief sets and that this unifier is not included in the restricted set. Then there must be some other mgu unifiers  $\sigma_1 \subseteq \sigma$  and  $\sigma_2 \subseteq \sigma$  which are equal to or more general than  $\sigma$ , which means  $\sigma \subseteq \sigma_1 \wedge \sigma \subseteq \sigma_2$  and  $p_1 \sigma_1 \in BB_1$  and  $p_2 \sigma_2 \in BB_2$ . However because  $\rho U_{BB_1}^{p_1}$  and  $\rho U_{BB_2}^{p_2}$  contains every mgu for individual predicates and belief bases, then both  $\sigma_1$  and  $\sigma_2$  belong to these sets and because  $\sigma \subseteq \sigma_1 \cup \sigma_2$  then either  $\sigma$  or a more general unifier  $\sigma$  must be included in the restriction.

From Theorem 1 it follows that if a restriction is applied to  $\rho U_{BB_1}^{p_1}$  and  $\rho U_{BB_2}^{p_2}$  then it results in every possible substitution that still fulfils both queries of  $p_1$  to  $BB_1$  and  $p_2$  to  $BB_2$ .

We will add one more lemma to this definition, which we will use later.

**Lemma 1:** If we obtain PUS by restriction

$$\rho U = \rho U_1 \sqcap \rho U_2,$$

then it is valid that

$$\forall \sigma \exists \sigma_1 (\sigma \in \rho U \wedge \sigma_1 \in \rho U_1 \wedge \sigma_1 \subseteq \sigma \wedge \sigma \in \{\sigma_1\} \sqcap \rho U_2)$$

**Proof:** We need to show that if a substitution arises by performing a restriction operation, then for

every substitution  $\sigma$  in this result there is at least one substitution in the first operand  $\sigma_1$ , that is a subset of  $\sigma$  such that the substitution  $\sigma$  would arise by performing the restriction even if the first operand were the set containing only that substitution  $\{\sigma_1\}$ . Since such a substitution must have arisen as a result of the merging operation  $\sigma = \sigma_1 \times \sigma_2$  and  $\sigma_1 \in \rho U_1$ . From Definition 4, this operation is performed as  $\sigma_1 \cup \sigma_2$  and it is clear that  $\sigma \subseteq \sigma_1 \cup \sigma_2$ . But then since  $\sigma_2 \in \rho U_2$  then  $\{\sigma_1\} \sqcap \rho U_2$  will also perform merging  $\sigma_1 \times \sigma_2$  and its result will also be part of the resulting set of substitutions.

An essential aspect of the BDI agents is that they build a hierarchy of plans. If the agent is about to go from one plan to another in the hierarchy, it must also transfer appropriate substitutions. For this purpose, we introduce the PUS intersection function.

**Definition 6:** The intersection is a function denoted by the symbol  $\sim$  and is defined for one PUS  $\rho U$  and two predicates  $p_1$  and  $p_2$  as follows:

$$p_1, \rho U \sim p_2 \stackrel{\text{def}}{=} \rho U(p_2, I\sigma(p_1, \rho U)) \quad (5)$$

**Definition 7:** The intersection function for two PUS  $\rho U_1$  and  $\rho U_2$  and two predicates  $p_1$  and  $p_2$  is defined as follows:

$$p_1, \rho U_1 \sim p_2, \rho U_2 \stackrel{\text{def}}{=} (p_1, \rho U_1 \sim p_2) \sqcap \rho U_2 \quad (6)$$

The last function which we define is called decision-making. Contrary to the previous operations and functions, this one is defined abstractly. Here we only declare its structure without any further specification on how to realize it.

**Definition 8:** Decision-making is a function denoted as  $Dec$  that for a PUS and a predicate  $p$  provides a ground substitution which is a subset of a substitution from the PUS.

When we write that  $Dec(\rho U, p) = \sigma$  then it must be valid that

$$\exists \sigma_1 (\sigma_1 \in \rho U, \sigma \subseteq \sigma_1, Var(p) = dom(\sigma)) \quad (7)$$

and  $\sigma$  is a ground substitution for  $p$ .

So the agent decides on some substitution from  $\rho U$  and from that determines the binding of the free variables in  $p$ . If this substitution is applied to  $p$ , then it makes  $p$  a ground predicate. But in addition, this substitution is based on those substitutions that the agent keeps as possible.

## 4. EXECUTING PLANS USING LATE BINDINGS

We have already mentioned that we will be working with a set of substitutions, i.e. some PUS that will be stored during the execution of the agent's plan. The plan becomes part of the agent's intent when the reasoning process chooses it for some event. At that same moment, the plan is assigned a PUS that the agent operates on while executing the plan. These substitutions are made by the agent in the course of its practical reasoning. So, unlike other AgentSpeak(L) interpreter systems, in our case the plan variables do not have to be replaced immediately, but can be kept separately as PUSs, which we will now call the plan context. This context changes as the agent performs actions and achieves goals from the plan body. If such a PUS is assigned to an event, we will speak of an event context. This will arise from the actual context of the plan that triggered the event, and we explain this in more detail below. Further in this text, we will talk about weak instances of plans and events when plans and events have contexts and other information associated with them according to the following definitions.

**Definition 9:** A weak plan instance is a triple  $\langle te, h, ctx \rangle$  containing a plan trigger event  $te$ , a plan body  $h = h_1; h_2; \dots; h_m$ , and a plan context  $ctx$ .

Similarly, we define weak event instances as tuples.

**Definition 10:** A weak event instance is a tuple  $\langle evt, ctx \rangle$  where  $evt$  is an event, and  $ctx$  is a context.

Its purpose is to represent an event that has arisen during the execution of a weak plan instance. It is necessary to have weak event instances because when an agent is pursuing a goal by a weakly instantiated plan, it need not know the exact substitution of the goal variables. Instead, the agent selects a single event with a context representing all possible goals that are sufficient to satisfy the declared goal. The agent decides how to proceed based on which goal or goals the agent subplan will achieve. It means, that it is sufficient to meet only one or a subset of these goals.

Furthermore, we will use the abbreviations WEI for weak event instances and WPI for weak plan instances.

## 4.1 Practical reasoning for weak instances

What differs from the original interpretation is the way the applicability and relevance are recognized. When there are WEIs and a plan, then a plan is relevant to the WEIs when its triggering event and the WEI event are unifiable concerning the WEI context. At the beginning, the plan has no context that would play a role in the unification process. Then it is enough that the PUS intersection of the triggering event, the goal event, and its context create a set containing at least one substitution. This can be seen as a query by triggering event to the base, which is formed as a set of instantiations from the WEI. Notice that it is valid also when the substitution is an empty set, i.e. if the resulting set contains a single element, namely the empty set. This is what would happen if the plan's triggering event were ground. Now let us define this 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 WEI  $\langle evt, ctx \rangle$  when  $(evt, ctx \sim te) \neq \{\}$

A plan is applicable when every of the plan's context conditions (here  $b_1 \dots b_n$ ) is satisfied in the actual state of the agent's belief base. This means that it is possible to find a PUS for every context condition and the belief base and consequently to make restrictions among them. From Theorem 1, it follows that if the restrictions result in a nonempty set, then there is at least one unifier for every context condition of the plan and an agent's belief in its belief base.

**Definition 12:** A plan  $te: b_1 \wedge b_2 \dots \wedge b_n \leftarrow h_1; h_2; \dots h_m$  is applicable in the actual agent's belief base BB, when  $\rho U(b_1, BB) \sqcap \dots \sqcap \rho U(b_n, BB) \neq \{\}$

Every plan which is both relevant and applicable can be considered as a means for the goal and the agent may choose it as its intended means. When the plan is both relevant and applicable, then the restriction of the PUS from Definition 11 and the PUS from Definition 12 must be also a non-empty PUS. Using the PUS intersection from Definition 7, we may write  $ctx_1 = ((evt, ctx \sim te) \sqcap \rho U(b_1, BB) \sqcap \dots \sqcap \rho U(b_n, BB)) \neq \{\}$  and then  $ctx_1$  is a context which, together with the plan's body  $h_1; h_2; \dots h_m$ , forms new WPI that can be an intended means for the WEI.

We can think of a WEI as a specification of more than one goal in the form of an event. By creating a set of instances for a given predicate in a WEI and for all substitutions from a given context, we obtain all event instances that represent all reachable goals

represented by that WEI. To satisfy a given WEI, it is sufficient for the agent to satisfy at least one of these goals. The following Lemma will show that for a WEI and a chosen plan, its default context contains all substitutions for which this plan is relevant and applicable to some WEI instance. In the remainder of this paper, when we refer to a WEI instance, we will mean a single instance created by applying the WEI context to a WEI predicate. By a WEI instance, we will then also mean one particular goal represented by this WEI.

**Lemma 2:** The result of practical reasoning for some WEI  $\langle evt, ctx \rangle$ , if a suitable plan is found, is a PUS that contains all the most general substitutions for which the chosen plan  $t_e: \Psi \leftarrow plan$  is relevant with respect to a given some event instance  $e \in I\sigma(evt, ctx)$  and applicable in the current state of the agent's belief base.

**Proof:** First consider a plan without the context conditions  $t_e \leftarrow plan$ . Then the result of practical reasoning is  $\Theta = (evt, ctx \sim t_e)$ . By Definition 6,  $(evt, ctx \sim t_e) = \rho U(t_e, I\sigma(evt, ctx))$  and further by Definition 1,  $\Theta = \{mgu(t_e, p'): p' \in I\sigma(evt, ctx)\}$ . Assume that there is a substitution  $\varphi \notin \Theta$ , for which this plan is relevant with respect to some  $e \in I\sigma(evt, ctx)$  and applicable in the agent's BB. Since, due to the absence of context, every relevant plan is also applicable, then suppose that  $\varphi$  is mgu for some  $t_e$  and an event instance  $e$ , but then  $\varphi \notin \{mgu(t_e, p): p \in I\sigma(evt, ctx)\}$  which is a contradiction. From a different perspective, we can view such inference as querying  $t_e$  into the BB containing all instances of a given WEI, and the result of this query is all possible mgu for which this query is satisfied. Thus, in the first stage of practical reasoning, we obtain all the substitutions that make this plan relevant to some instance of the event. All these substitutions are mgu for all event instances of the WEI and the plan's triggering event. Assume, that the context were non-empty  $\Psi = b_1 \wedge \dots \wedge b_n$  and we have  $\varphi$  which is a substitution for which the plan is relevant and applicable. Suppose now that  $\varphi \notin \Xi$  where  $\Xi$  is calculated, as we showed above, as intersection by  $((evt, ctx \sim t_e) \sqcap \rho U(b_1, BB) \sqcap \dots \sqcap \rho U(b_n, BB))$  and we know that  $\Theta = (evt, ctx \sim t_e)$  provides all mgu for  $t_e$  and the instance of WEI under consideration,  $\varphi$  must also be a superset of some such mgu of  $\Theta$ . Furthermore, if we consider the first condition  $b_1$  of the context condition  $\Psi$  and perform  $\Theta \sqcap, \rho U(b_1, BB)$ , then by Theorem 1 we obtain all substitutions for which they are valid both with respect to the relevance of the plan and with respect to the context condition  $b_1$ . Similarly, for  $\Theta \sqcap, \rho U(b_1, BB), \dots \sqcap \rho U(b_n, BB)$ , the result is all substitutions that make the plan relevant and all

context conditions are satisfied, so the condition  $\varphi$  must be present in  $\Xi$ .

Thanks to Lemma 2, we know that if a plan is adopted and a WPI containing that plan and some context is created, then that context contains all possible substitutions for which the execution of that plan is applicable and, in particular, valid against some WEI for which it was chosen. Again, this can be looked at from a different perspective. The selection of the appropriate plan can be seen as a successful query in which we use the plan's triggering event and event instance set. The choice of the applicable plan is then seen as a series of queries for each part of the context conditions to the agent's BB. The resulting substitutions according to Theorem 1 contain all substitutions that satisfy all these queries.

Now we want to find out all the substitutions that, on the one hand, still satisfy that the plan is a relevant means for them to achieve some event instance, but also make all the acts of the plan, including possible subplans, feasible. We will do this sequentially.

## 4.2 Plan body execution

We start with the simplest example where the plan does not contain any act, and choosing it would already result in the event being fulfilled. Then the substitutions found during the practical reasoning provide the goals that have been achieved.

**Example 1:** For some WEI, where the event predicate is  $!transport\_means(X, Y, M)$ . is to represent the location X from which someone or something is to be moved to location Y and M is the means of transport and the WEI context may be  $\{\{[berlin/X], [prague/X]\}\}$ . The relevant plan is for example  $+!transport\_means(X, Y, M): get\_tm(X, M)$  and this contains only one context condition and if the agent's belief base contains for example  $get\_tm(prague, car)$ .  $get\_tm(berlin, airplane)$ .  $get\_tm(berlin, train)$ .  $get\_tm(berlin, bus)$ . then for the given WEI the initial and resulting context is  $\{\{[berlin/X], [airplane/M]\}, \{[berlin/X], [train/M]\}, \{[prague/X], [car/M]\}\}$ .

In the following paragraphs, our focus will be on how the WPI changes from the moment of its creation to the moment of its successful execution. We won't even go into how the agent responds to situations where the WPI fails. This would occur when the context in the WPI transforms to an empty set. However, we will assume that each execution of an act in the body of the WPI plan will result in the transformation of its context into another context that is not an empty set. We will show specifically how

this context will be transformed, one by one, by the possible acts that the agent can execute.

First, we define the test goals and their interpretation.

**Definition 13:** A test goal is executed for some WPI  $\langle t_e, ?p(\mathbf{t}); planbody, ctx \rangle$  and if successfully executed, then the new WPI is

$$\langle t_e, planbody, ctx \sqcap \rho U(p(\mathbf{t}), BB) \rangle$$

By executing one test goal  $?p_i(\mathbf{t})$  the context is changed to include all substitutions that match all previous queries and must now include this new query. Let us denote the result of the testing goal by  $\rho U$  and this contains all mgu for  $p_i(\mathbf{t})$  and some belief from the agent's current BB. If we perform  $ctx' = ctx \sqcap \rho U$ , then we get a new context  $ctx'$ , which, by Theorem 1, contains all substitutions that are valid for all queries performed so far (including those performed during practical inference).

**Definition 14:** Performing actions either external or internal transforms the context from WPI  $\langle t_e, a(\mathbf{t}); planbody, ctx \rangle$  so that the new WPI is  $\langle t_e, planbody, ctx \sqcap \{Dec(ctx, a(\mathbf{t}))\} \rangle$

The agent can create a ground predicate for the action before executing it. Thus, all free variables must be bound to a specific atom. This is the case when the *Dec* function of Definition 8 is used. This is a situation that causes non-determinism in the agent's behavior and we do not provide a concrete implementation of it in this text. Anyway, the agent performs the action described as  $a(\mathbf{t})$  where  $\mathbf{t}$  are terms (in our case we consider atoms or variables). After this action is performed, all variables from the terms  $\mathbf{t}$  are uniquely bound to the same variables in all context substitutions.

**Example 2:** For an action  $go(A, B)$  with context  $ctx = \{\{[paris/A], [car/B], [mon/X]\}, \{[rio/A], [plane/B], [tue/X]\}, \{[brno/A], [bicycle/B], [std/X]\}, \{[brno/A], [bicycle/B], [sun/X]\}\}$  the agent can decide to perform one of  $go(paris, car)$ ,  $go(paris, plane)$  or  $go(brno, bicycle)$ . The first action is created by performing the first substitution on the original action, the second action by performing the second, and the third action by performing the third or fourth. The resulting contexts are then  $ctx = \{\{[paris/A], [car/B], [mon/X]\}$  for the first action,  $ctx = \{\{[rio/A], [plane/B], [tue/X]\}$  for the second action, and  $ctx = \{\{[brno/A], [bicycle/B], [std/X]\}, \{[brno/A], [bicycle/B], [sun/X]\}\}$  for the third action.

The result of performing an action is to reduce the context to only those substitutions that are consistent with the action performed and the agent loses those

options that do not match the action performed. From the above, we summarize the following Lemma. Its proof follows from the above definitions and their analysis.

**Lemma 3:** Executing a plan. which contains only actions and test goals (i.e., does not invoke subplans by the act of achievement goal) the original WPI is transformed into another whose context contains all the substitutions for which this one is relevant to the original WEI that invoked it, was applicable with respect to the specified context conditions, and corresponds to all the executed acts of this plan.

Before we move on to the implementation of the subplans, let's take a closer look at the relationship between the resulting context and the WEI that triggered the plan. We want to show that it is true that if the execution of a WPI succeeded, then for some event that can be instantiated from the predicate and context in that WEI, the plan would also succeed, it means that our approach sounds.

**Lemma 4:** If the execution of a plan invoked by a WEI succeeds, then the execution of the plan would also succeed for at least one event instance from the WEI if used alone.

**Proof:** The context is changed only by using the restriction operation in the case of both types of acts (testing, action). According to Lemma 1, we know that each substitution thus produced is a non-strict superset of a substitution from the previous context. The second operand in the restriction is given by the act performed. Since non-strict superset is a transitive relation, then every substitution in the resulting context is an non-strict superset to some mgu arising during the practical reasoning, which we denote by  $\Theta$  in the proof of Lemma 2. Thus, if the context changes from  $\Theta$  to  $\Xi = ctx_0$  and then by carrying out the acts of the plans to  $ctx_1, ctx_2 \dots ctx_n$ , then by Lemma 1 for each  $\sigma_n \in ctx_n$  there is a sequence  $\sigma_r \in \Theta$ ,  $\sigma_0 \in ctx_0, \sigma_1 \in ctx_1 \dots \sigma_{n-1} \in ctx_{n-1}$  such that  $\sigma_r \subseteq \sigma_0 \subseteq \sigma_1 \subseteq \dots \sigma_{n-1} \subseteq \sigma_n$ . Furthermore, thanks to the same Lemma, this sequence would be replayed using the same actions, according to Definition 13 and Definition 14. That is, if some one mgu from  $\Theta$  were taken during practical reasoning, above  $\sigma_r$ , then the plan would be feasible.

A corollary of this Lemma is that if the plan is successfully executed, the analysis of the resulting substitutions can determine for which all mgu of  $\Theta$  the plan would be successfully executable. A simple way to determine this is that if an element from  $\Theta$  is a subset of some element from the resulting context, then this element determines the goal that will be achieved by the plan, since it is the mgu obtained

during practical reasoning in the search for a relevant plan for some WEI.

**Example 3:** For a WEI  $< !go(X, Y, Z), \{[paris/X], [rio/X]\} >$  the plan with triggering event  $+!go(A, B, C)$  is chosen. The plan is relevant and the  $\Theta = \{[paris/A], [rio/A]\}$  contains all the mgu for the WEI and triggering event of the plan. The resulting context after such a plan is executed is for example  $\{[paris/A], [car/B], [mon/C], [paris/A], [train/B], [tue/C]\}$ . We can see, that this plan is feasible for the goal  $go(paris, Y, Z)$  because  $\{[paris/A]\} \subseteq \{[paris/A], [car/B], [mon/C]\}$  and  $\{[paris/A]\} \subseteq \{[paris/A], [train/B], [tue/C]\}$ , but is not feasible for  $go(rio, Y, Z)$ . Moreover, we know that this plan could have been feasible if we had driven on Monday, or taken the train on Tuesday.

The above example shows that plan execution could be used to find the answer as understood in (Sardina & Padgham, 2011). The answer-seeking principle is familiar from the execution of programs in PROLOG, and the process of querying and obtaining answers can be seen in the execution of the goals specified in the agent's plans. The query is given in the form of a predicate that need not be ground, and the answers are possibly bindings to variables for which such a predicate is valid in the database. Since we have shown that the resulting context contains all substitutions for which a plan is feasible for some WEI, these substitutions are also all possible answers to the queries, which can be considered as all the entities of a given WEI. The example above shows that these answers could be reached through the trigger event and final context of the plan. Creating the instance set from them creates a base, which we query with a predicate from the WEI. Thus, in Example 3, such a base would contain two predicates  $go(paris, car, mon)$  and  $go(paris, train, tue)$  which we obtained by applying substitutions  $\{[paris/A], [car/B], [mon/C]\}$  and  $\{[paris/A], [train/B], [tue/C]\}$  to the trigger event  $go(A, B, C)$ . The question would be  $go(X, Y, Z)$ , which will provide answers  $\{[paris/X], [car/Y], [mon/Z]\}, \{[paris/X], [train/Y], [tue/Z]\}$ .

We summarize the above in the following Lemma, which we provide without proof, noting that we have demonstrated its validity above.

**Lemma 5:** For some WEI and for a selected relevant and applied plan containing only actions or test goals, the interpreted system interpreting the program with late bindings will provide all possible answers with respect to the chosen actions.

We can now consider plans that contain some achievement goals, and show that what is stated in the previous Lemma holds for them as well. But first we need how the achievement goal is implemented.



**Definition 15:** Achievemnt goal is executed for some WPI  $\langle t_e, !p(\mathbf{t}); \text{planbody}, \text{ctx} \rangle$  such that WEI  $\langle !p(\mathbf{t}), \text{ctx}' \rangle$  where  $\text{ctx}'_1 = \text{ctx} \langle p(\mathbf{t}) \rangle$  is generated, (see Definition 3). The execution of the WPI is suspended until the sub-plan is successfully executed, which ends up as  $\langle !p(\mathbf{t}_2), \text{null}, \text{ctx}_{2f} \rangle$ . The original WPI then continues as

$$\langle t_e, \text{planbody}, (p(\mathbf{t}_2), \text{ctx}_{2f} \sim p(\mathbf{t}), \text{ctx}) \rangle$$

The change of context here is done according to Definitions 6 and 7, by which we can break this operation down as

$$p(\mathbf{t}_2), \text{ctx}_{2f} \sim p(\mathbf{t}), \text{ctx} = \left( p(\mathbf{t}_2), \text{ctx}_{2f} \sim p(\mathbf{t}) \right) \sqcap \text{ctx} = \rho U \left( p(\mathbf{t}), \text{I}\sigma(p(\mathbf{t}_2), \text{ctx}_{2f}) \right) \sqcap \text{ctx}.$$

Thus, the query  $p(\mathbf{t})$  to the basis  $\text{I}\sigma(p(\mathbf{t}_2), \text{ctx}_{2f})$  is performed as a broad unification using the predicate of the original achievement goal  $!p(\mathbf{t})$ . This query provides all answers according to the successfully executed chosen plan. These answers are then restricted by the context of the original plan and the result is the new context of the original WPI.

**Example 4:** Suppose that there is a WPI

$$\langle !\text{visit}(V, X), !\text{go}(X, Y, Z); \text{planbody}, \{ \{ \text{paris}/X, \text{uncle}/V \}, \{ \text{rio}/X, \text{friend}/V \} \} \rangle$$

which was generated for this WEI. Such a plan could be implemented for an agent who wants to go on a trip and visit someone at the same time. The context of this plan gives the possibilities of visiting an uncle in Paris, or a friend in Rio. The achievement goal  $!go(X, Y, Z)$  is to find a means of getting to some place the agent would like to be. This plan in Example 3 ended with the agent receiving the answers  $\{ \{ \text{paris}/X, \text{car}/Y, \text{mon}/Z \}, \{ \text{paris}/X, \text{train}/Y, \text{tue}/Z \} \}$ . after obtaining all possible answers according to this plan. This is the result of the operation  $p(\text{go}(A, B, C)), \{ \{ \text{paris}/A, \text{car}/B, \text{mon}/C \}, \{ \text{paris}/A, \text{train}/B, \text{tue}/C \} \} \sim \text{go}(X, Y, Z)$ . By Definition 15, the restriction with the original WPI context which was  $\{ \{ \text{paris}/X, \text{uncle}/V \}, \{ \text{rio}/X, \text{friend}/V \} \}$  is to be made and then the new context is  $\{ \{ \text{paris}/X, \text{car}/Y, \text{mon}/Z, \text{uncle}/V \}, \{ \text{paris}/X, \text{train}/Y, \text{tue}/Z, \text{uncle}/V \} \}$ .

**Theorem 2:** If for some WEI  $\langle \text{evt}, \text{ctx} \rangle$  a plan is successfully executed, and the resulting WPI of this plan is  $\langle t_e, \text{null}, \text{ctx}_f \rangle$ , then the result of executing the achievement goals according to the predicate  $\text{evt}$  and the substitutions in the context of  $\text{ctx}$  is a result encompassing all the goals achieved by the plan with respect to the selected actions and subplans.

**Proof:** Thanks to the above Lemma 5, we know that if we have a WEI of the form  $\langle \text{evt}, \text{ctx} \rangle$  and a WPI of some plan that does not contain an achievement goal is successfully executed, then all possible answers to the given goals that can be achieved by executing this WPI are obtained. Thus, if we have an achievement goal  $!p(\mathbf{t})$  within some plan, then a WEI  $\langle !p(\mathbf{t}), \text{ctx}' \rangle$  where  $\text{ctx}'$  is the actual context is created, and if this goal was achieved by the WPI of some plan without other achievement goals, then the agent would receive all possible answers to  $!p(\mathbf{t})$ .

According to the definition of the implementation of this type of goals, we know that the new context is given by these (all) answers and restrictions with the original context and thus, according to Theorem 1, contains all possible substitutions that agree with the original context and the answers received. Such a WPI, if successfully executed, will provide through the resulting context all possible substitutions that agree with the executed plans. By induction, if an agent would need more immersions in subplans for some goal, say up to level  $n$ , then we can prove that WPIs that need at most two levels of subplans achieve all possible substitutions, and then proceed similarly up to level  $n$ .

## 5. CONCLUSIONS

In this paper, we have presented an interpretation of the execution of a plan written in AgentSpeak(L) using late binding of variables. What this approach brings to the interpretation of this language is that the agent thus has available at any point in time all the substitutions that are valid with respect to all the acts it has performed with a given WPI. This in itself does not necessarily improve the rationality of the agent's behaviour. Still, at some points the agent has to decide for some substitutions when it has to perform an action. However, late binding of variables can help avoid premature erroneous choices of substitutions, because by delaying their selection, the agent has the opportunity to evaluate the situation when it must make the decision. Moreover, if the agent maintains all the ways it can still bind variables, this can be beneficial even when the agent is pursuing multiple goals and can look for synergies between plans for these goals, but also for synchronizing the actions of agents in a multi-agent group.

Systems that have addressed this so far either do not work with variables at all, or consider variable binding as an alternative when the initially chosen binding fails (Sardina & Padgham, 2011) (Dastani, 2008). When researchers do work with focuses and with the choice of plans for goals, they currently work

with goal plan trees, in which formulas are written as propositional and thus do not consider predicates as such (Yao & Logan, Action-Level Intention Selection for BDI Agents, 2016) (Yao, Logan, & Thangarajah, Robust execution of BDI agent programs by exploiting synergies between intentions, 2016). We believe that a system that can handle multiple options at once by storing possible bindings until a decision must be made will yield an increase in agent behaviour in these respects. However, these are suggestions for our future research, in which we intend to use the principle of late bindings introduced in this paper.

## ACKNOWLEDGEMENTS

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

## REFERENCES

- Bratman, M. E. (1987). *Intention, Plans and Practical Reason*. Harvard University Press.
- Bratman, M. E., Israel, D. J., & Pollack, M. E. (1988). Plans and resource-bounded practical reasoning. *Philosophy and AI: Essays at the interface*, stránky 7-22.
- Cohen, P. R., & Levesque, H. J. (1990). Intention is choice with commitment. *Artificial Intelligence, Volume 42, Issues 2-3*, stránky 213-261.
- Dastani, M. (2008). 2APL: a practical agent programming language. *Autonomous Agents and Multi-Agent Systems volume 16*, stránky 214-248.
- d'Iverno, M., Kinny, D., Luck, M., & Wooldridge, M. (1998). A Formal Specification of dMARS. *Lecture Notes in AI, 1365*, stránky 155-176.
- Georgeff, M. P., & Lansky, A. L. (1987). Reactive reasoning and planning. *Proceedings of the Sixth National Conference on Artificial Intelligence (AAAI-87)*, stránky 198-204.
- Rao, A. S. (1996). AgentSpeak(L): BDI agents speak out in a logical computable language. *MAAMAW 1996: Agents Breaking Away*, stránky 42-55.
- Rao, A. S., & Georgeff, M. (1991). Modeling Rational Agents within a BDI-Architecture. *Australian Artificial Intelligence Institute*.
- Sardina, S., & Padgham, L. (2011). A BDI agent programming language with failure handling, declarative goals, and planning. *Autonomous Agents and Multi-Agent Systems 23*, stránky 18-70.
- Winikoff, M. (2005). An AgentSpeak Meta-Interpreter and Its Applications. *Proceedings of the Third international conference on Programming Multi-Agent Systems*, stránky 123-138.
- Yao, Y., & Logan, B. (2016). Action-Level Intention Selection for BDI Agents. *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*.
- Yao, Y., Logan, B., & Thangarajah, J. (2016). Robust execution of BDI agent programs by exploiting synergies between intentions. *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)*, (stránky 2558-2564).
- Zboril, F., Koci, R., Janousek, V., & Mazal, Z. (2008). Reactive Planning with Weak Plan Instances. *Proceedings of ISDA 2008*.
- Zboril, F., Zboril, F. V., & Kral, J. (2013). Flexible Plan Handling using Extended Environment. *12th International Scientific Conference on Informatics*.