Subgoal Semantics in Agent Programming

M. Birna van Riemsdijk Mehdi Dastani John-Jules Ch. Meyer

ICS, Utrecht University, The Netherlands
{birna, mehdi, jj}@cs.uu.nl

Abstract. This paper investigates the notion of subgoals as used in plans in cognitive agent programming languages. These subgoals form an abstract representation of more concrete courses of action or plans. Subgoals can have a procedural interpretation (directly linked to a concrete plan) or a declarative one (the state to be reached as represented by the subgoal is taken into account). We propose a formal semantics for subgoals that interprets these declaratively, and study the relation between this semantics and the procedural subgoal semantics of the cognitive agent programming language 3APL. We prove that subgoals of 3APL can be programmed to behave declaratively, although the semantics is defined procedurally.

1 Introduction

This paper presents an *observation* about the cognitive agent programming language 3APL [8]. The observation is related to the notion of a *goal*. This is an important concept in cognitive agent programming languages. Goals are introduced to specify an agent's proactive behavior. Many languages and platforms have been proposed to implement (represent and process) an agent's goals [11,8,15,4,16,3,14,2]. The way in which goals are dealt with varies from language to language. In some programming languages goals are interpreted in a *procedural* way as processes that need to be executed. In others goals are interpreted in a *declarative* way as states to be reached. Yet other languages combine both aspects. Procedural goals are also often called *plans*, which is a terminology we will also use in this paper.

While the procedural interpretation might arguably be considered more standard, the declarative interpretation of goals also has several advantages. Most importantly in this context, is the fact that declarative goals provide for the possibility to decouple plan execution (i.e., the execution of a procedural goal) and goal achievement (i.e., the achievement of a declarative goal) [16]. If a plan fails, the goal that was to be achieved by the plan remains a goal of the agent. The agent can then for example select a different plan or wait for the circumstances to change for the better.[1]

A common usage of goals, and the one we are concerned with in this paper, is that of *subgoals* as occurring in the plans of the agent.[2] These plans are often

[1] See e.g. [13] for a more elaborate discussion on the advantages of declarative goals.

[2] A usage of the term subgoal that we do not consider in this paper is usage in the logical sense, where for example p is considered to be a subgoal of the goal $p \wedge q$ [13].

built from basic actions which can be executed directly, and subgoals which can be viewed as representing a course of action in a more abstract way. An agent can for example have the plan to go to the bus stop, to take the bus into town,[3] and then to achieve the goal of buying a birthday cake. This goal of buying a birthday cake will have to be fulfilled by selecting a more concrete plan of for example which shop to go to, etc.

Just as goals in general, subgoals of plans can also be categorized as either procedural or declarative. In the procedural interpretation, subgoals are linked directly to plans. Their only role is the abstract representation of a more concrete plan. In the declarative interpretation, the fact of whether the state that is represented by the subgoal is achieved (for example through the execution of a corresponding concrete plan), is somehow taken into account. In the birthday cake example, this means that it is important whether the execution of the concrete plan of which shop to go to etc., has resulted in a state in which the birthday cake is actually bought. If it turns out that the goal of buying the cake is not reached after having gone to the specific shop, the agent could select another plan to try a different shop. A declarative interpretation of subgoals could yield more flexible agent behavior, because of the decoupling between plan execution and goal achievement.

We thus argue that it is important to be able to express a declarative notion of subgoals in a cognitive agent programming language. This paper aims to investigate whether these declarative subgoals can be expressed in the language 3APL. In order to do this, we first make precise what we mean exactly by declarative subgoals, by defining a simple formal semantics for subgoals that interprets these in a declarative way (sections 2 and 3). We then compare this semantics with the semantics of 3APL (section 4). We show that 3APL has a notion of subgoal, but it is a procedural kind of subgoal. It turns out, however, that although subgoals of 3APL are defined to have a procedural semantics, a 3APL agent can nevertheless be *programmed* to have these subgoals behave *declaratively*. This observation (and a formal proof that it is correct) is the main contribution of this paper.

The 3APL language family [8,15,4] is an example of a set of languages in which subgoals are interpreted procedurally. Languages and platforms from the AgentSpeak family [9,5,11,1,6] also have a procedural view on subgoals, although the mechanism differs from that of 3APL. We conjecture that a similar result for an implementation of declarative subgoals can be obtained for AgentSpeak, although this is left for future research. An example of a declarative view on subgoals is the high-level language of Winikoff et al. [16]. The declarative semantics we propose in section 3 is comparable with that of [16], although [16] has a much more elaborate plan language. An elaborate plan language is however not needed for the purpose of comparison with the procedural subgoals of 3APL. Establishing a formal relation with the work of Winikoff et al. is left for

[3] Assuming that both going to the bus stop and taking the bus into town are actions that can be executed directly.

future research. The Jadex platform [2] incorporates declarative and procedural interpretations, although the platform does not have a formal semantics.

2 Syntax

In this section and the next, we present the syntax and semantics of a simple programming language with plans containing subgoals that have a declarative interpretation. Throughout this paper, we assume a language of propositional logic $\mathcal{L}$ with negation and conjunction and based on a set of atoms Atom. The symbol $\models$ will be used to denote the standard entailment relation for $\mathcal{L}$. Below, we define the language of plans. A plan is a sequence of basic actions and statements of the form $achieve(p)$ (subgoals), where $p \in \mathsf{Atom}$. Informally, basic actions can change the beliefs of an agent if executed, and a statement of the form $achieve(p)$ means that p should be achieved, before the agent can continue the execution of the rest of the plan.

Definition 1 *(plans)* Let $\mathsf{BasicAction}$ with typical element a be the set of basic actions and let $p \in \mathsf{Atom}$. The set of plans Plan with typical element π is then defined as follows.

$$\pi ::= a \mid achieve(p) \mid \pi_1; \pi_2$$

We use ϵ to denote the empty plan and identify $\epsilon; \pi$ and $\pi; \epsilon$ with π.

We use a simple plan language, focused on subgoals. The language could however be extended to include, e.g., test and non-deterministic choice. Also, the subgoals could be extended to arbitrary formulas, rather than just atoms. For atomic subgoals however, a correspondence with the procedural goals of 3APL can be established. For arbitrary subgoals this cannot be done in the general case, as we conjecture.

In order to define the plans that can be used for achieving the subgoals, we use so-called plan generation rules. Informally, a plan generation rule $p \Rightarrow \pi$ specifies that the plan π can be selected to try to achieve the subgoal p. One could add a condition on the beliefs of the agent to the rule, specifying that the rule can only be applied if the agent has a certain belief. We however leave this out for reasons of simplicity.

Definition 2 *(plan generation rule)* The set of plan generation rules $\mathcal{R}_{\mathsf{PG}}$ is defined as follows: $\mathcal{R}_{\mathsf{PG}} = \{p \Rightarrow \pi \mid p \in \mathsf{Atom}, \pi \in \mathsf{Plan}\}$.

An agent in this paper is a tuple, consisting of an initial belief base (a consistent set of formulas from $\mathcal{L}$ representing what the agent believes about the world), an initial plan, a set of plan generation rules and a belief update function $\mathcal{T}$. This function $\mathcal{T}$, taking a basic action and a belief base and yielding a new belief base, is used to define how belief bases are updated if a basic action is executed. The function is undefined for a basic action and belief base, if the basic action cannot be executed on this belief base. This function is used for technical convenience and is not further specified, as it is not needed for the purpose of this paper.

Definition 3 *(subgoal achievement agent)* Let $\Sigma = \{\sigma \mid \sigma \subseteq \mathcal{L}, \sigma \not\models \bot\}$ be the set of belief bases. A subgoal achievement agent, typically denoted by $\mathcal{A}$, is a tuple $\langle \sigma, \pi, \mathsf{PG}, \mathcal{T} \rangle$ where $\sigma \in \Sigma$ is the belief base, $\pi \in \mathsf{Plan}$ is the initial plan, and $\mathsf{PG} \subseteq \mathcal{R}_{\mathsf{PG}}$ is a set of plan generation rules. $\mathcal{T}$ is a partial function of type $(\mathsf{BasicAction} \times \Sigma) \rightarrow \Sigma$.

When defining the semantics of plan execution, we use the notion of a configuration. A configuration consists of a belief base and a plan, which are the elements of an agent that can change during its execution.

Definition 4 *(configuration)* A configuration is a pair $\langle \sigma, \pi \rangle$ where $\sigma \in \Sigma$ and $\pi \in \mathsf{Plan}$.

3 Semantics

In this section, we provide a semantics for the execution of plans containing subgoals, that interprets these declaratively. We define the semantics using a transition system [10]. A transition system for a programming language consists of a set of axioms and derivation rules for deriving transitions for this language. A transition is a transformation of one configuration into another and it corresponds to a single computation step. Let $\mathcal{A} = \langle \sigma, \pi, \mathsf{PG}, \mathcal{T} \rangle$ be a subgoal achievement agent. The transition system $\mathsf{Trans}_{\mathcal{A}}$ for this agent is then given by the definitions below.

A basic action at the head of a plan can be executed in a configuration if the function $\mathcal{T}$ is defined for this action and the belief base in the configuration. The execution results in a change of belief base as specified through $\mathcal{T}$, and the action is removed from the plan.

Definition 5 *(action execution)* Let $a \in \mathsf{BasicAction}$.

$$\frac{\mathcal{T}(a, \sigma) = \sigma'}{\langle \sigma, a; \pi \rangle \rightarrow \langle \sigma', \pi \rangle}$$

The following two definitions specify the possible transitions in case a statement of the form $achieve(p)$ is the first "action" of the plan. Both transitions rely upon a declarative interpretation of p, as it is checked whether p is believed to be reached. Definition 6 gives the transition in case p is achieved. The statement $achieve(p)$ is then removed from the plan.

Definition 6 *(subgoal achievement)*

$$\frac{\sigma \models p}{\langle \sigma, achieve(p); \pi \rangle \rightarrow \langle \sigma, \pi \rangle}$$

The next transition rule specifies the transition for an $achieve(p)$ statement in case p is not achieved. In this case, a plan should be generated in order to achieve p. This can be done if there is a plan generation rule of the form $p \Rightarrow \pi'$ in the rule base of the agent. The transition that can then be derived, specifies that the plan π' is placed at the head of the plan.

Definition 7 *(plan generation)* Let $p \Rightarrow \pi' \in \mathsf{PG}$.

$$\frac{\sigma \not\models p}{\langle \sigma, achieve(p); \pi \rangle \rightarrow \langle \sigma, \pi'; achieve(p); \pi \rangle}$$

It is important to note that the statement $achieve(p)$ is not removed from the plan if a plan generation rule is applied. If π' is executed and p is still not derivable from the agent's beliefs (p is not reached), a different rule with p as the head could be applied (if it exists), to achieve p by other means. In any case, a statement $achieve(p)$ will not be removed from the plan if p is not reached.

Given the transition system $\mathsf{Trans}_{\mathcal{A}}$ for subgoal achievement agent $\mathcal{A}$ as specified above, one can construct computation runs for $\mathcal{A}$. A computation run is a sequence of configurations, such that each consecutive configuration can be obtained from the previous through the application of a transition rule. The initial configuration of the computation run is formed by the initial belief base and plan of $\mathcal{A}$. A successful computation run is a run of which the final configuration has an empty plan. The semantics of $\mathcal{A}$ is then defined as the set of successful computation runs of $\mathcal{A}$.

Definition 8 *(semantics of a subgoal achievement agent)* Let $\mathcal{A} = \langle \sigma_0, \pi_0, \mathsf{PG}, \mathcal{T} \rangle$ be a subgoal achievement agent. Let a computation run be a sequence of configurations. A successful computation run of agent $\mathcal{A}$ is a computation run $\langle \sigma_0, \pi_0 \rangle, \ldots, \langle \sigma_n, \epsilon \rangle$, such that $\forall_{1 \leq i \leq n} : \langle \sigma_{i-1}, \pi_{i-1} \rangle \rightarrow \langle \sigma_i, \pi_i \rangle$ is a transition that can be derived in $\mathsf{Trans}_{\mathcal{A}}$. The semantics of $\mathcal{A}$ is the set $\{\theta \mid \theta \text{ is a successful computation run of } \mathcal{A}\}$.

A property of the semantics that reflects that subgoals are interpreted declaratively, is the following: if a plan of the form $achieve(p)$ is the initial plan of the agent, then it holds for any successful computation run of this agent ending in some belief base σ_n, that p follows from σ_n.

Proposition 1 Let $\mathcal{A} = \langle \sigma_0, \pi_0, \mathsf{PG}, \mathcal{T} \rangle$ be a subgoal achievement agent, and let $\theta = \langle \sigma_0, \pi_0 \rangle, \ldots, \langle \sigma_n, \epsilon \rangle$ be a successful computation run of $\mathcal{A}$. If π_0 is of the form $achieve(p)$, we have that $\sigma_n \models p$.

At this point we remark that the semantics of our *achieve* statement is closely related to the "bringing it about" operator as introduced by Segerberg [12] in the area of philosophical logic. His operator δ satisfies the property $[\delta p]p$ (expressed in a kind of dynamic logic), which would in our notation be the property $[achieve(p)]p$, stating that p always holds after the "execution" of $achieve(p)$. This is a reformulation of the above proposition in dynamic logic. A formal study of the relation of our work with that of Segerberg is left for future research.

4 Comparison with 3APL

4.1 Syntax and Semantics

In this section, we present a propositional and otherwise slightly simplified version of 3APL [8]. We introduce these simplification for reasons of clarity and

simplicity, but they are not fundamental. 3APL is similar to the language as presented in section 2. A 3APL agent has a belief base (set of formulas from $\mathcal{L}$), a plan, a set of so-called plan revision rules for manipulating its plan, and a belief update function $\mathcal{T}$.

The language of plans of 3APL agents is similar to the plan language of definition 1. A 3APL plan however does not contain *achieve* statements. The 3APL counterpart of these subgoals is called an *achievement goal* [8], and was dubbed *abstract plan* in later papers [15,4]. An abstract plan is basically a string, just as a basic action is a string (but as we will see, abstract plans have a different semantics). For the comparison with subgoal achievement agents however, we take the set of abstract plans as consisting not of an arbitrary set of strings, but of exactly the atoms of $\mathcal{L}$. Further, we add the possibility to test whether an atom follows from the belief base or not, and we add non-deterministic choice.

Definition 9 *(3APL plans)* Let $\mathsf{BasicAction}$ with typical element a be the set of basic actions and let $\mathsf{AbstractPlan}$ with typical element p be the set of abstract plans, such that $\mathsf{AbstractPlan} = \mathsf{Atom}$ and $\mathsf{AbstractPlan} \cap \mathsf{BasicAction} = \emptyset$. The set of 3APL plans Plan' with typical element π is then defined as follows.

$$\pi ::= a \mid p \mid p? \mid \neg p? \mid \pi_1; \pi_2 \mid \pi_1 + \pi_2$$

We use ϵ to denote the empty plan and identify $\epsilon; \pi$ and $\pi; \epsilon$ with π.

Abstract plans obtain their meaning through the plan revision rules of the 3APL agent. These rules have a plan as the head and as the body. During execution of a plan, a plan revision rule can be used to replace a prefix of the plan, which is identical to the head of the rule, by the plan in the body. If the agent for example executes a plan $a; b; c$ and has a plan revision rule $a; b \Rightarrow d$, it can apply this rule, yielding the plan $d; c$. Here we do not use the general plan revision rules that can have a composed plan as the head. We only use rules with an abstract plan as the head and a plan as the body.

Definition 10 *(plan revision rule)* The set of plan revision rules $\mathcal{R}_{\mathsf{PR}}$ is defined as follows: $\mathcal{R}_{\mathsf{PR}} = \{p \Rightarrow \pi \mid p \in \mathsf{AbstractPlan}, \pi \in \mathsf{Plan}'\}$.

Plan revision rules thus very much resemble the plan generation rules of definition 2 (syntactically, that is), but the body is a 3APL plan, i.e., a plan from Plan'. As we will explain shortly, the *semantics* of plan revision rules however differs from that of plan generation rules in important ways.

The semantics of 3APL agents is defined by means of a transition system, as given below. The first transition rule is used to derive a transition for action execution, and is similar to the transition rule of this kind for subgoal achievement agents (definition 5). The second transition specifies the application of a plan revision rule of the form $p \Rightarrow \pi'$ to a plan of the form $p; \pi$. If the rule is applied, the abstract plan p is replaced by the body of the rule, yielding the plan $\pi'; \pi$. It is important to note that it is *not* tested whether p holds, and further that p is *replaced* by π', rather than yielding the plan $\pi'; p; \pi$.

The transition rules for test and non-deterministic choice are fairly standard. Note however that a test for $\neg p$ succeeds if it is *not* the case that p follows from the belief base, rather than having this test succeed if $\neg p$ *does* follow. The reason for this choice should become clear in the sequel. Further, some transitions are labelled with i, which we will also need in the sequel.

Definition 11 *(3APL transition system)* A 3APL agent $\mathcal{A}'$ is a tuple $\langle \sigma, \pi, \mathsf{PR}, \mathcal{T} \rangle$, where $\sigma \in \Sigma$, $\pi \in \mathsf{Plan}'$, $\mathsf{PR} \subseteq \mathcal{R}_{\mathsf{PR}}$ and $\mathcal{T}$ as in definition 3. The transition system $\mathsf{Trans}_{\mathcal{A}'}$ for this 3APL agent is then defined as follows, where $a \in \mathsf{BasicAction}$ and $p \Rightarrow \pi' \in \mathsf{PR}$.

$$1)\ \frac{\mathcal{T}(a,\sigma) = \sigma'}{\langle \sigma, a; \pi \rangle \rightarrow \langle \sigma', \pi \rangle} \qquad 2)\ \frac{}{\langle \sigma, p; \pi \rangle \rightarrow_i \langle \sigma, \pi'; \pi \rangle}$$

$$3)\ \frac{\sigma \models p}{\langle \sigma, p?; \pi \rangle \rightarrow_i \langle \sigma, \pi \rangle} \qquad 4)\ \frac{\sigma \not\models p}{\langle \sigma, \neg p?; \pi \rangle \rightarrow_i \langle \sigma, \pi \rangle}$$

$$5)\ \frac{\langle \sigma, \pi_1 \rangle \rightarrow \langle \sigma', \pi_1' \rangle}{\langle \sigma, (\pi_1 + \pi_2); \pi \rangle \rightarrow \langle \sigma', \pi_1'; \pi \rangle} \qquad 6)\ \frac{\langle \sigma, \pi_2 \rangle \rightarrow \langle \sigma', \pi_2' \rangle}{\langle \sigma, (\pi_1 + \pi_2); \pi \rangle \rightarrow \langle \sigma', \pi_2'; \pi \rangle}$$

Before we move on to formally investigating the relation between 3APL and subgoal achievement agents, we elaborate on the notion of an abstract plan or achievement goal as used in 3APL. Hindriks et al. [8] remark the following with respect to achievement goals:

> *Achievement goals are atomic propositions from the logical language $\mathcal{L}$. The use of atoms as achievement goals, however, is very different from the use of atoms as beliefs. Whereas in the latter case atoms are used to represent and therefore are of a declarative nature, in the former case they serve as an abstraction mechanism like procedures in imperative programming and have a procedural meaning.*

Hindriks et al. thus take the set of achievement goals/abstract plans to be the atoms from $\mathcal{L}$ (a first order language in their case). Then they remark that although achievement goals are atoms, they do *not* have a declarative interpretation. The fact that an achievement goal is an atom and could thus in principle be tested for example against the belief base, is not used in defining its semantics. The language of achievement goals could thus have been any language of strings (which is in fact the approach of later papers [15,4]). Hindriks et al. [8] however do remark the following with respect to a possible assertional reading of achievement goals:[4]

[4] As a first order language is used in [8], the original text states $p(\vec{t})$ instead of p. This is a predicate name parameterized with a sequence of terms.

Apart from the procedural reading of these goals, however, an assertional reading is also possible. An achievement goal p would then be interpreted as specifying a goal to achieve a state of affairs such that p. We think such a reading is valid in case the plans for achieving an achievement goal p actually do establish p.

The *"plans for achieving an achievement goal p"* are the plans as specified through the plan revision rules, i.e., a plan revision rule $p \Rightarrow \pi$ specifies that π is a plan for achieving p. According to Hindriks et al., this assertional or declarative reading of achievement goals is thus *only* valid under the strong requirement that π actually reaches p. This is thus in contrast with the semantics for subgoals as we have introduced, as these subgoals are by definition interpreted in a declarative manner.

4.2 3APL and Subgoals

We will show in this section that, although the semantics of plan generation rules and plan revision rules differ in important ways, it *is* possible to define a mapping from an arbitrary subgoal achievement agent to a 3APL agent, such that the 3APL agent "simulates" the behavior of the subgoal achievement agent. In the sequel, the plan π_s denotes π in which all occurrences of statement of the form $achieve(p)$ are replaced with p.

We first remark that the naive translation, in which a plan generation rule $p \Rightarrow \pi$ is translated to a plan revision rule $p \Rightarrow \pi_s$, does not do the trick. The reason that this translation does not work, is precisely the difference of interpretation between *achieve* statements and abstract plans, i.e., declarative versus procedural. If an abstract plan p occurs at the head of a plan $p; \pi$, the plan revision rule $p \Rightarrow \pi'$ can be applied, yielding $\pi'; \pi$. After the execution of π', the plan π will be executed, *regardless* of whether p is actually achieved at that point. In the case of a plan $achieve(p); \pi$ of a subgoal achievement agent, the plan generation rule can be applied (if p is not believed), yielding the plan $\pi'; achieve(p); \pi$. After the execution of π', the agent will test whether p is achieved. If it is achieved, it will continue with the execution of π. If however p is *not* achieved, it will apply a rule once more to generate a plan to achieve p. It is nevertheless important to mention that *if* it is the case that π' actually establishes p (and this holds for all plan generation rules), it *can* be proven that the 3APL agent as obtained through this naive translation, simulates the subgoal achievement agent. For reasons of space, we however omit this proof.

Translation We now turn to the translation of a subgoal achievement agent into a 3APL agent, for which it holds that the 3APL agent as obtained in this way, simulates the subgoal achievement agent. As would be expected, the important part of the translation is the mapping of plan generation rules onto plan revision rules.

Definition 12 *(transformation of subgoal achievement agent into 3APL agent)* Let $s : \mathsf{Plan} \rightarrow \mathsf{Plan}'$ be a function that takes a plan π of a subgoal achievement

agent (definition 1), and yields this plan in which all statements of the form *achieve*(p) are replaced by p, thus yielding a plan in Plan' (definition 9). We will in the sequel use the notation π_s for $s(\pi)$.

The function $t : \mathcal{R}_{\mathsf{PG}} \rightarrow \mathcal{R}_{\mathsf{PR}}$, taking a plan generation rule and yielding a corresponding plan revision rule, is then defined as follows.

$$t(p \Rightarrow \pi) \;=\; p \Rightarrow ((\neg p?; \pi_s; p) + p?)$$

The function t is lifted to sets of plan generation rules in the obvious way.

Let $\mathcal{A} = \langle \sigma, \pi, \mathsf{PG}, \mathcal{T} \rangle$ be a subgoal achievement agent. The 3APL agent corresponding with $\mathcal{A}$ is then $\langle \sigma, \pi_s, t(\mathsf{PG}), \mathcal{T} \rangle$. Finally, we define a function τ that takes a configuration from the transition system of $\mathcal{A}$ of the form $\langle \sigma, \pi \rangle$, and yields the configuration $\langle \sigma, \pi_s \rangle$.

Informally, this mapping can be used to obtain a 3APL agent that simulates a subgoal achievement agent, because of the following. Consider a 3APL agent with the plan $p; \pi$, and the plan revision rule $p \Rightarrow ((\neg p?; \pi'_s; p) + p?)$ as obtained from the plan generation rule $p \Rightarrow \pi'$. This plan revision rule can then be applied to this plan (regardless of whether p is believed or not), yielding the plan $((\neg p?; \pi'_s; p) + p?); \pi$.

Now assume that p is believed. In that case, the test $p?$ succeeds and $\neg p?$ fails, which means that the plan in the next configuration will have to be π. This thus implements that p is skipped if believed to be achieved, which corresponds with the semantics of the statement *achieve*(p).

Now assume that p is not believed. In that case, the plan in the next configuration will have to be $\pi'_s; p; \pi$. This corresponds with the semantics of *achieve*(p) in case p is not believed: the plan π' is placed at the head of the plan, not replacing the *achieve*(p) statement. After the execution of π'_s, we are left with the plan $p; \pi$. The plan revision rule $p \Rightarrow ((\neg p?; \pi'_s; p) + p?)$ (or a different rule with p as the head) will then be applied again. If p is achieved, the agent will continue with the execution of π as explained. If p is not achieved, the mechanism as just described will be set in motion. All this thus corresponds with the behavior of *achieve* statements in the subgoal achievement agent.

Bisimulation Theorem We now move on to formally establishing this correspondence. For this, we introduce the notion of a translation bisimulation as used in [7, Chapter 8] (slightly adapted). Informally, a translation bisimulation translates an agent from a so-called source language to an agent from the target language that "can do the same things". In our case, we translate subgoal achievement agents to 3APL agents.

We have to show that for each transition in the transition system for a subgoal achievement agent $\mathcal{A}$, there is a corresponding transition in the transition system for the corresponding 3APL agent $\mathcal{A}'$. This "transition" in $\mathsf{Trans}_{\mathcal{A}'}$, actually does not have to be a single transition, but may consist of a number of so-called idle transitions, and one non-idle transition. The idle transitions in $\mathsf{Trans}_{\mathcal{A}'}$ are those labelled with i. Intuitively, these idle transitions form implementation details of

$\mathcal{A}'$, and do not have to be matched by a transition of $\mathcal{A}$.[5] In the sequel, the transition relations of $\mathcal{A}$ and $\mathcal{A}'$ will respectively be denoted by $\rightarrow^{\mathcal{A}}$ and $\rightarrow^{\mathcal{A}'}$, and $\rightarrow_i^{\mathcal{A}'}$ denotes the restriction of $\mathcal{A}'$ to idle transitions.

The new transition relation that abstracts from idle steps is denoted by $\rightarrow_*^{\mathcal{A}'}$. It only exists for $\mathsf{Trans}_{\mathcal{A}'}$, as $\mathsf{Trans}_{\mathcal{A}}$ does not contain idle steps. It is defined as follows, where d_j with $1 \leq j \leq n$ are configurations derivable in $\mathsf{Trans}_{\mathcal{A}'}$: $d_1 \rightarrow_*^{\mathcal{A}'} d_n$ iff there is a (possibly empty) series of idle transitions $d_1 \rightarrow_i^{\mathcal{A}'} d_2 \rightarrow_i^{\mathcal{A}'} \ldots \rightarrow_i^{\mathcal{A}'} d_{n-1}$ and a single non-idle transition $d_{n-1} \rightarrow^{\mathcal{A}'} d_n$.

If we can show that for each transition in the transition system for a subgoal achievement agent $\mathcal{A}$, there is a corresponding transition in the transition system for the corresponding 3APL agent $\mathcal{A}'$, we will have established that $\mathcal{A}'$ generates *at least* the behavior of $\mathcal{A}$. In order to establish that $\mathcal{A}'$ does not generate any (alternative) behavior not having a counterpart in $\mathcal{A}$, we also have to show that any non-idle transition of $\mathcal{A}'$ corresponds with a transition of $\mathcal{A}$. A transition $c_1 \rightarrow^{\mathcal{A}} c_2$ corresponds with a transition $d_1 \rightarrow^{\mathcal{A}'} d_2$ or $d_1 \rightarrow^* d_2$ iff $d_1 = \tau(c_1)$ and $d_2 = \tau(c_2)$.

The result can only be proven if we assume that at least one plan generation rule of the form $p \Rightarrow \pi$ exists for every $p \in \mathsf{Atom}$. If this would not be the case, there would be a mismatch: a statement $achieve(p)$ could be removed from a plan if p holds (without there being a plan generation rule for p), but an abstract plan p can only be "removed" if first a plan revision rule is applied.

Theorem 1 *(translation bisimulation)* Let $\mathcal{A} = \langle \sigma, \pi, \mathsf{PG}, \mathcal{T} \rangle$ be a subgoal achievement agent such that for each $p \in \mathsf{Atom}$ there is at least one rule of the form $p \Rightarrow \pi$ in PG, and let $\mathcal{A}' = \langle \sigma, \pi_s, t(\mathsf{PG}), \mathcal{T} \rangle$ be the corresponding 3APL agent. We then have that for every configuration c_1 of $\mathcal{A}$, $d_1 = \tau(c_1)$ implies the following:

1. If $c_1 \rightarrow^{\mathcal{A}} c_2$, then $d_1 \rightarrow_*^{\mathcal{A}'} d_2$, such that $d_2 = \tau(c_2)$.
2. If $d_1 \rightarrow^{\mathcal{A}'} d_2$, then for some c_2, $c_1 \rightarrow^{\mathcal{A}} c_2$, such that $d_2 = \tau(c_2)$.

Proof: 1. We have to show that for every transition $c_1 \rightarrow^{\mathcal{A}} c_2$ in $\mathsf{Trans}_{\mathcal{A}}$, there is a corresponding (sequence of) transition(s) $d_1 \rightarrow_*^{\mathcal{A}'} d_2$ in $\mathsf{Trans}_{\mathcal{A}'}$ such that $d_2 = \tau(c_2)$.

Let $\langle \sigma, a; \pi \rangle \rightarrow^{\mathcal{A}} \langle \sigma', \pi \rangle$ be a transition as derived through the transition rule of definition 5. We then have that the transition $\langle \sigma, a; \pi_s \rangle \rightarrow^{\mathcal{A}'} \langle \sigma', \pi_s \rangle$ can be derived in $\mathsf{Trans}_{\mathcal{A}'}$, by means of the first transition rule. We also have that $\langle \sigma', \pi_s \rangle = \tau(\langle \sigma', \pi \rangle)$, yielding the desired result for action execution transitions.

Let $\langle \sigma, achieve(p); \pi \rangle \rightarrow^{\mathcal{A}} \langle \sigma, \pi \rangle$ be a transition as derived through the transition rule of definition 6, which means that $\sigma \models p$ has to hold. Let $p \Rightarrow \pi'$ be a

[5] The choice of idle transitions for 3APL might seem strange, as the application of a plan revision rule is an idle transition, whereas the non-deterministic choice is not, although the latter might seem an implementation detail, rather than the former. The reason is, that the application of a plan revision rule cannot be matched directly with a transition of a goal achievement agent, whereas the particular usage of non-deterministic choice, as specified through the translation, *can*.

plan generation rule of $\mathcal{A}$ (a rule of this form has to exist by assumption). We then have, because $t(p \Rightarrow \pi')$ is a plan revision rule of $\mathcal{A}'$, that the transitions

$$\langle \sigma, p; \pi_s \rangle \rightarrow_i^{\mathcal{A}'} \langle \sigma, ((\neg p?; \pi'_s; p) + p?); \pi_s) \rangle \rightarrow^{\mathcal{A}'} \langle \sigma, \pi_s \rangle$$

can be derived in $\mathsf{Trans}_{\mathcal{A}'}$, by means of the transition rule for plan revision and those for test and non-deterministic choice. We also have that $\langle \sigma, \pi_s \rangle = \tau(\langle \sigma, \pi \rangle)$, yielding the desired result for subgoal achievement transitions.

Let $\langle \sigma, achieve(p); \pi \rangle \rightarrow^{\mathcal{A}} \langle \sigma, \pi'; achieve(p); \pi \rangle$ be a transition as derived through the transition rule of definition 7, which means that $\sigma \not\models p$ has to hold, and $p \Rightarrow \pi'$ has to be a plan generation rule in PG. We then have that the transitions

$$\langle \sigma, p; \pi_s \rangle \rightarrow_i^{\mathcal{A}'} \langle \sigma, ((\neg p?; \pi'_s; p) + p?); \pi_s) \rangle \rightarrow^{\mathcal{A}'} \langle \sigma, \pi'_s; p; \pi_s \rangle$$

can be derived in $\mathsf{Trans}_{\mathcal{A}'}$, by means of the transition rule for plan revision and those for test and non-deterministic choice. We also have that $\langle \sigma, \pi'_s; p; \pi_s \rangle = \tau(\langle \sigma, \pi'; achieve(p); \pi \rangle)$, yielding the desired result for plan generation transitions. We have shown the desired result for every transition $c_1 \rightarrow^{\mathcal{A}} c_2$, thereby proving 1. For reasons of space, we omit the proof of 2. □

5 Conclusion

In this paper, we have studied the relation between declarative and procedural interpretations of subgoals as occurring in the plans of cognitive agents. In particular, we have compared our definition of declaratively interpreted subgoals with the semantics of the procedurally interpreted achievement goals in the language 3APL. As we have shown, it is possible to obtain a 3APL agent that simulates the behavior of the subgoal achievement agent, by translating plan generation rules to plan revision rules in a specific way.

Future research will address the relation between our subgoal achievement agents and AgentSpeak(L), and with the work of Winikoff et al. [16]. Also, we will investigate other semantics for subgoals, e.g., semantics making use of a goal base.

To the best of our knowledge, this is the first time that a correspondence between declarative and procedural subgoals is investigated and established. We believe that the investigations as described in this paper shed some light on the expressiveness of languages with procedural goals, and that this is an important piece of the puzzle of the incorporation of declarative goals in cognitive agent programming languages.

References

1. R. H. Bordini and A. F. Moreira. Proving the asymmetry thesis principles for a BDI agent-oriented programming language. *Electronic Notes in Theoretical Computer Science*, 70(5), 2002. http://www.elsevier.nl/gejng/31/29/23/125/23/29/70.5.008.pdf.

2. L. Braubach, A. Pokahr, D. Moldt, and W. Lamersdorf. Goal representation for BDI agent systems. In *Programming multiagent systems, second international workshop (ProMAS'04)*, volume 3346 of *LNAI*, pages 44–65. Springer, Berlin, 2005.
3. M. Dastani and L. van der Torre. Programming BOID-Plan agents: deliberating about conflicts among defeasible mental attitudes and plans. In *Proceedings of the Third Conference on Autonomous Agents and Multi-agent Systems (AAMAS'04)*, pages 706–713, New York, USA, 2004.
4. M. Dastani, M. B. van Riemsdijk, F. Dignum, and J.-J. Ch. Meyer. A programming language for cognitive agents: goal directed 3APL. In *Programming multiagent systems, first international workshop (ProMAS'03)*, volume 3067 of *LNAI*, pages 111–130. Springer, Berlin, 2004.
5. M. d'Inverno, D. Kinny, M. Luck, and M. Wooldridge. A formal specification of dmars. In *ATAL '97: Proceedings of the 4th International Workshop on Intelligent Agents IV, Agent Theories, Architectures, and Languages*, pages 155–176, London, UK, 1998. Springer-Verlag.
6. R. Evertsz, M. Fletcher, R. Jones, J. Jarvis, J. Brusey, and S. Dance. Implementing industrial multi-agent systems using JACK™. In *Proceedings of the first international workshop on programming multiagent systems (ProMAS'03)*, volume 3067 of *LNAI*, pages 18–49. Springer, Berlin, 2004.
7. K. V. Hindriks. *Agent programming languages - programming with mental models.* PhD thesis, 2001.
8. K. V. Hindriks, F. S. de Boer, W. van der Hoek, and J.-J. Ch. Meyer. Agent programming in 3APL. *Int. J. of Autonomous Agents and Multi-Agent Systems*, 2(4):357–401, 1999.
9. F. F. Ingrand, M. P. Georgeff, and A. S. Rao. An architecture for real-time reasoning and system control. *IEEE Expert: Intelligent Systems and Their Applications*, 7(6):34–44, 1992.
10. G. D. Plotkin. A Structural Approach to Operational Semantics. Technical Report DAIMI FN-19, University of Aarhus, 1981.
11. A. S. Rao. AgentSpeak(L): BDI agents speak out in a logical computable language. In W. van der Velde and J. Perram, editors, *Agents Breaking Away (LNAI 1038)*, pages 42–55. Springer-Verlag, 1996.
12. K. Segerberg. Bringing it about. *Journal of Philosophical Logic*, 18:327–347, 1989.
13. M. B. van Riemsdijk, M. Dastani, F. Dignum, and J.-J. Ch. Meyer. Dynamics of declarative goals in agent programming. In J. A. Leite, A. Omicini, P. Torroni, and P. Yolum, editors, *Proceedings of the second international workshop on Declarative agent languages and technologies (DALT'04)*, pages 17–32, 2004.
14. M. B. van Riemsdijk, M. Dastani, and J.-J. Ch. Meyer. Semantics of declarative goals in agent programming. In *Proceedings of the fourth international joint conference on autonomous agents and multiagent systems (AAMAS'05)*, Utrecht, 2005. To appear.
15. M. B. van Riemsdijk, W. van der Hoek, and J.-J. Ch. Meyer. Agent programming in Dribble: from beliefs to goals using plans. In *Proceedings of the second international joint conference on autonomous agents and multiagent systems (AAMAS'03)*, pages 393–400, Melbourne, 2003.
16. M. Winikoff, L. Padgham, J. Harland, and J. Thangarajah. Declarative and procedural goals in intelligent agent systems. In *Proceedings of the eighth international conference on principles of knowledge respresentation and reasoning (KR2002)*, Toulouse, 2002.