Agent Programming with Declarative Goals

Koen V. Hindriks, Frank S. de Boer,
Wiebe van der Hoek, and John-Jules Ch. Meyer

April 17, 2000

Abstract

A long and lasting problem in agent research has been to close the gap between agent logics and agent programming frameworks. The main reason for this problem of establishing a link between agent logics and agent programming frameworks is identified and explained by the fact that agent programming frameworks have not incorporated the concept of a declarative goal. Instead, such frameworks have focused mainly on plans or goals-to-do instead of the end goals to be realised which are also called goals-to-be. In this paper, a new programming language called GOAL is introduced which incorporates such declarative goals. The notion of a commitment strategy - one of the main theoretical insights due to agent logics, which explains the relation between beliefs and goals - is used to construct a computational semantics for GOAL. Finally, a proof theory for proving properties of GOAL agents is presented. The programming logic for GOAL is a temporal logic extended with belief and goal modalities. An example program is proven correct by using this programming logic.

1 Goal-Oriented Agent Programming

In the early days of agent research, an attempt was made to make the concept of agents more precise by means of logical systems. This effort resulted in a number of - mainly - modal logics for the specification of agents which formally defined notions like belief, goal, intention, etc. associated with agents [14, 19, 3, 4]. The relation of these logics with more practical approaches remains unclear, however, to this day. Several efforts to bridge this gap have been attempted. In particular, a number of agent programming languages have been developed to bridge the gap between theory and practice [13, 9]. These languages show a clear family resemblance with one of the first agent programming languages Agent-0 [17, 6], and also with the language ConGolog [5, 8, 7].

These programming languages define agents in terms of their corresponding beliefs, goals, plans and capabilities. Although they define similar notions as in the logical approaches, there is one notable difference. In logical approaches, a goal is a declarative concept, whereas in the cited programming languages goals are defined as sequences of actions or plans. The terminology used differs from case to case. However, whether they are called commitments (Agent-0), intentions (AgentSpeak [13]), or goals (ZAPL [10]) makes little difference: all these notions are structures built from actions and therefore similar in nature to plans. With respect to ConGolog, a more traditional computer science perspective is adopted, and the corresponding structures are simply called programs. The PLAC language [18], a successor of AGENT0, also focuses more on extending AGENT0 to a language with complex planning structures (which are not part of the programming language itself!) than on providing a clear theory of declarative goals of agents as part of a programming language and in this respect is similar to AgentSpeak and ZAPL. The type of goal included in these languages may also be called a goal-to-do and provides for a kind of procedural perspective on goals.

In contrast, a declarative perspective on goals in agent languages is still missing. Because of this mismatch it has not been possible so far to use modal logics which include both belief and goal modalities for the specification and verification of programs written in such agent languages and it has been impossible to close the gap between agent logics and programming frameworks so
far. The value of adding declarative goals to agent programming lies both in the fact that it offers a new abstraction mechanism as well as that agent programs with declarative goals more closely approximate the intuitive concept of an intelligent agent. To fully realise the potential of the notion of an intelligent agent, a declarative notion of a goal, therefore, should also be incorporated into agent programming languages. In this paper, we introduce the agent programming language GOAL, which takes the declarative concept of a goal seriously and which provides a concrete proposal to bridge the gap between theory and practice. We offer a complete theory of agent programming in the sense that our theory provides both for a programming framework and a programming logic for such agents. In contrast with other attempts [17, 21] to bridge the gap, our programming language and programming logic are related by means of a formal semantics. Only by providing such a formal relation it is possible to make sure that statements proven in the logic concern properties of the agent.

2 The Programming Language GOAL

In this section, we introduce the programming language GOAL (for Goal-Oriented Agent Language). The programming language GOAL is inspired by work in concurrent programming, in particular by the language UNITY designed by Chandy and Misra [2]. The basic idea is that a set of actions which execute in parallel constitutes a program. However, whereas UNITY is a language based on assignment to variables, the language GOAL is an agent-oriented programming language that incorporates more complex notions such as belief, goal, and agent capabilities which operate on high-level information instead of simple values.

As in most agent programming languages, GOAL agents select actions on the basis of their current mental state. A mental state consists of the beliefs and goals of the agent. However, in contrast to most agent languages, GOAL incorporates a declarative notion of a goal that is used by the agent to decide what to do. Both the beliefs and the goals are drawn from one and the same logical language, \( \mathcal{L} \), with associated consequence relation \( \models \). An agent thus keeps two databases, respectively called the belief base and the goal base. The difference between these two databases originates from the different meaning assigned to sentences stored in the belief base and sentences stored in the goal base. To clarify the interaction between beliefs and goals, one of the more important problems that needs to be solved is establishing a meaningful relationship between beliefs and goals. This problem is solved here by imposing a constraint on mental states that is derived from the default commitment strategy that agents use. The notion of a commitment strategy is explained in more detail below. The constraint imposed on mental states requires that an agent does not believe that \( \phi \) is the case if it has a goal to achieve \( \phi \), and, moreover, requires \( \phi \) to be consistent if \( \phi \) is a goal.

Definition 2.1 (mental state)
A mental state of an agent is a pair \( (\sigma, \gamma) \) where \( \sigma \subseteq \mathcal{L} \) are the agent’s beliefs and \( \gamma \subseteq \mathcal{L} \) are the agent’s goals and \( \sigma \) and \( \gamma \) are such that for any \( \phi \in \gamma \) we have:

- \( \phi \) is not entailed by the agent’s beliefs \( (\sigma \not\models \phi) \),
- \( \phi \) is consistent \( (\not\models \neg \phi) \), and
- \( \sigma \) is consistent \( (\sigma \not\models \text{false}) \).

A mental state does not contain a program or plan component in the ‘classical’ sense. Although both the beliefs and the goals of an agent are drawn from the same logical language, as we will see below, the formal meaning of beliefs and goals is very different. This difference in meaning reflects the different features of the beliefs and the goals of an agent. The declarative goals are best thought of as achievement goals in this paper. That is, these goals describe a goal state that the agent desires to reach. Mainly due to the temporal features of such goals many properties of beliefs fail for goals. For example, the fact that an agent has the goal to be at home and the goal to be at the movies does not allow the conclusion that this agent also has the conjunctive goal
to be at home and at the movies at the same time. As a consequence, less stringent consistency requirements are imposed on goals than on beliefs. An agent may have the goal to be at home and the goal to be at the movies simultaneously; assuming these two goals cannot consistently be achieved at the same time does not mean that an agent cannot have adopted both in the language GOAL.

In this paper, we assume that the language $\mathcal{L}$ used for representing beliefs and goals is a simple propositional language. As a consequence, we do not discuss the use of variables nor parameter mechanisms. Our motivation for this assumption is the fact that we want to present our main ideas in their simplest form and do not want to clutter the definitions below with details. Also, more research is needed to extend the programming language with a parameter passing mechanism, and to extend the programming logic for GOAL with first order features.

The language $\mathcal{L}$ for representing beliefs and goals is extended to a new language $\mathcal{L}_M$ which enables us to formulate conditions on the mental state of an agent. The language $\mathcal{L}_M$ consists of so called mental state formulas. A mental state formula is a boolean combination of the basic mental state formulas $B_\phi$, which expresses that $\phi$ is believed to be the case, and $G_\phi$, which expresses that $\phi$ is a goal of the agent.

**Definition 2.2 (mental state formula)**
The set of mental state formulas $\mathcal{L}_M$ is defined by:

- if $\phi \in \mathcal{L}$, then $B_\phi \in \mathcal{L}_M$,
- if $\phi \in \mathcal{L}$, then $G_\phi \in \mathcal{L}_M$,
- if $\varphi_1, \varphi_2 \in \mathcal{L}_M$, then $\neg \varphi_1, \varphi_1 \land \varphi_2 \in \mathcal{L}_M$.

The usual abbreviations for the propositional operators $\lor, \rightarrow, \leftrightarrow$ are used. We write true as an abbreviation for $B(p \lor \neg p)$ for some $p$ and false for $\neg \text{true}$.

A third basic concept in GOAL is that of an agent capability. The capabilities of an agent consist of a set of so called basic actions. The effects of executing such a basic action are reflected in the beliefs of the agent and therefore a basic action is taken to be a belief update on the agent’s beliefs. A basic action thus is a mental state transformer. Two examples of agent capabilities are the actions $\text{ins}(\phi)$ for inserting $\phi$ in the belief base and $\text{del}(\phi)$ for removing $\phi$ from the belief base. Agent capabilities are not supposed to change the goals of an agent, but because of the constraints on mental states they may as a side effect modify the current goals. For the purpose of modifying the goals of the agent, two special actions $\text{adopt}(\phi)$ and $\text{drop}(\phi)$ are introduced to respectively adopt a new goal or drop some old goals. We write $\text{Bcap}$ and use it to denote the set of all belief update capabilities of an agent. $\text{Bcap}$ thus does not include the two special actions for goal updating $\text{adopt}(\phi)$ and $\text{drop}(\phi)$. The set of all capabilities is then defined as $\text{Cap} = \text{Bcap} \cup \{\text{adopt}(\phi),\text{drop}(\phi) \mid \phi \in \mathcal{L}\}$. Individual capabilities are denoted by $a$.

The set of basic actions or capabilities associated with an agent determines what an agent is able to do. It does not specify when such a capability should be exercised and when performing a basic action is to the agent’s advantage. To specify such conditions, the notion of a conditional action is introduced. A conditional action consists of a mental state condition expressed by a mental state formula and a basic action. The mental state condition of a conditional action states the conditions that must hold for the action to be selected. Conditional actions are denoted by the symbol $b$ throughout this paper.

**Definition 2.3 (conditional action)**
A conditional action is a pair $\varphi \rightarrow \text{do}(a)$ such that $\varphi \in \mathcal{L}_M$ and $a \in \text{Cap}$.

Informally, a conditional action $\varphi \rightarrow \text{do}(a)$ means that if the mental condition $\varphi$ holds, then the agent may consider doing basic action $a$. Of course, if the mental state condition holds in the current state, the action $a$ can only be successfully executed if the action is enabled, that is, only if its preconditions hold.

A GOAL agent consists of a specification of an initial mental state and a set of conditional actions.
Definition 2.4 (GOAL agent)
A GOAL agent is a triple $(\Pi, \sigma_0, \gamma_0)$ where $\Pi$ is a non-empty set of conditional actions, and $\langle \sigma_0, \gamma_0 \rangle$ is the initial mental state.

2.1 The Operational Semantics of GOAL

One of the key ideas in the semantics of GOAL is to incorporate into the semantics a particular commitment strategy (cf. [15, 3]). The semantics is based on a particularly simple and transparent commitment strategy, called blind commitment. An agent that acts according to a blind commitment strategy drops a goal if and only if it believes that that goal has been achieved. By incorporating this commitment strategy into the semantics of GOAL, a default commitment strategy is built into agents. It is, however, only a default strategy and a programmer can overwrite this default strategy by means of the drop action. It is not possible, however, to adopt a goal $\phi$ in case the agent believes that $\phi$ is already achieved.

The semantics of action execution should now be defined in conformance with this basic commitment principle. Recall that the basic capabilities of an agent were interpreted as belief updates. Because of the default commitment strategy, there is a relation between beliefs and goals, however, and we should extend the belief update associated with a capability to a mental state transformer that updates beliefs as well as goals according to the blind commitment strategy. To get started, we thus assume that some specification of the belief update semantics of all capabilities - except for the two special actions adopt and drop which only update goals - is given. Our task is, then, to construct a mental state transformer semantics from this specification for each action. That is, we must specify how a basic action updates the complete current mental state of an agent starting with a specification of the belief update associated with the capability only.

From the default blind commitment strategy, we conclude that if a basic action $a$ - different from an adopt or drop action - is executed, then a goal is dropped only if the agent believes that the goal has been accomplished after doing $a$. The revision of goals thus is based on the beliefs of the agent. The beliefs of an agent represent all the information that is available to an agent to decide whether or not to drop or adopt a goal. So, in case the agent believes that a goal has been achieved by performing some action, then this goal must be removed from the current goals of the agent. Besides the default commitment strategy, only the two special actions adopt and drop can result in a change to the goal base.

The initial specification of the belief updates associated with the capabilities $\text{Bcap}$ is formally represented by a partial function $T$ of type: $\text{Bcap} \times \wp(\mathcal{L}) \to \wp(\mathcal{L})$. $T(a, \sigma)$ returns the result of updating belief base $\sigma$ by performing action $a$. The fact that $T$ is a partial function represents the fact that an action may not be enabled or executable in some belief states. The mental state transformer function $\mathcal{M}$ is derived from the semantic function $T$ and also is a partial function. As explained, $\mathcal{M}(a, (\sigma, \gamma))$ removes any goal from the goal base $\gamma$ that have been achieved by doing $a$. The function $\mathcal{M}$ also defines the semantics of the two special actions adopt and drop. An adopt($\phi$) action adds $\phi$ to the goal base if $\phi$ is consistent and $\phi$ is not believed to be the case. A drop($\phi$) action removes every goal that entails $\phi$ from the goal base. As an example, consider the two extreme cases: drop($\neg \phi$) removes no goals, whereas drop($\text{false}$) removes all current goals.

Definition 2.5 (mental state transformer $\mathcal{M}$)

Let $(\sigma, \gamma)$ be a mental state, and $T$ be a partial function that associates belief updates with agent capabilities. Then the partial function $\mathcal{M}$ is defined by:

$$\mathcal{M}(a, (\sigma, \gamma)) = \{ \psi \in \gamma \mid T(a, \sigma) = \psi \} \text{ for } a \in \text{Bcap} \text{ if } T(a, \sigma) \text{ is defined},$$

$$\mathcal{M}(a, (\sigma, \gamma)) \text{ is undefined for } a \in \text{Bcap} \text{ if } T(a, \sigma) \text{ is undefined},$$

$$\mathcal{M}(\text{drop}(\phi), (\sigma, \gamma)) = \langle \sigma, \gamma \setminus \{ \psi \mid \psi = \phi \} \rangle,$$

$$\mathcal{M}(\text{adopt}(\phi), (\sigma, \gamma)) = \langle \sigma, \gamma \cup \{ \phi \} \rangle \text{ if } \sigma \not\equiv \phi \text{ and } \not\equiv \neg \phi,$$

$$\mathcal{M}(\text{adopt}(\phi), (\sigma, \gamma)) \text{ is undefined if } \sigma \equiv \phi \text{ or } \equiv \neg \phi.$$

The semantic function $\mathcal{M}$ maps an agent capability and a mental state to a new mental state. The capabilities of an agent are thus interpreted as mental state transformers by $\mathcal{M}$. Although it
is not allowed to adopt a goal $\phi$ that is inconsistent - an $\text{adopt}(\text{false})$ is not enabled - there is no check on the global consistency of the goal base of an agent built into the semantics. This means that it is allowed to adopt a new goal which is inconsistent with another goal present in the goal base. For example, if the current goal base $\gamma = \{p\}$ contains $p$, it is legal to execute the action $\text{adopt}(\neg p)$ resulting in a new goal base $\{p, \neg p\}$. Although inconsistent goals cannot be achieved at the same time, they may be achieved in some temporal order. Individual goals in the goal base, however, are required to be consistent. Thus, whereas local consistency is required (i.e. individual goals must be consistent), global consistency of the goal base is not required (i.e. $\gamma = \{p, \neg p\}$ is a legal goal base).

The second idea incorporated into the semantics concerns the selection of conditional actions. A conditional action $\varphi \rightarrow \text{do}(a)$ may specify conditions on the beliefs as well as conditions on the goals of an agent. As is usual, conditions on the beliefs are taken as a precondition for action execution: only if the agent’s current beliefs entail the belief conditions associated with $\varphi$ the agent will select $a$ for execution. The goal condition, however, is used in a different way. It is used as a means for the agent to determine whether or not the action will help bring about a particular goal of the agent. In short, the goal condition specifies where the action is good for. This does not mean that the action necessarily establishes the goal immediately, but rather may be taken as an indication that the action is helpful in bringing about a particular state of affairs. To make this discussion more precise, we introduce a formal definition of a formula $\phi$ that partially fulfills a goal in a mental state $\langle \sigma, \gamma \rangle$.

**Definition 2.6 (\phi partially fulfills a goal in a mental state)**

Let $\langle \sigma, \gamma \rangle$ be a mental state, and $\phi \in \mathcal{L}$. Then:

$$\phi \sim_{\sigma} \gamma \text{ iff for some } \psi \in \gamma : \psi \models \phi \text{ and } \sigma \not\models \phi$$

Informally, the definition of $\phi \sim_{\sigma} \gamma$ can be paraphrased as follows: the agent needs to establish $\phi$ to realise one of its goals in $\gamma$, but does not believe that $\phi$ is the case. The formal definition of $\phi \sim_{\sigma} \gamma$ entails that the realisation of $\phi$ would bring about at least part of one of the goals in the goal base $\gamma$ of the agent. The condition that $\phi$ is not entailed by the beliefs of the agent ensures that a goal is not a tautology. Of course, variations on this definition of the semantics of goals are conceivable. For example, one could propose a stronger definition of $\sim_{\sigma}$ such that $\phi$ brings about the complete realisation of a goal in the current goal base $\gamma$ instead of just part of such a goal. However, our definition of $\sim_{\sigma}$ provides for a simple and clear principle for action selection: the action in a conditional action is only executed in case the goal condition associated with that action partially fulfills some goal in the current goal base of the agent.

The semantics of belief conditions $\mathcal{B}\phi$, goal conditions $\mathcal{G}\phi$ and mental state formulas is defined in terms of the consequence relation $\models$ and the partially fulfills relation $\sim_{\sigma}$.

**Definition 2.7 (semantics of mental state formulas)**

Let $\langle \sigma, \gamma \rangle$ be a mental state.

- $\langle \sigma, \gamma \rangle \models \mathcal{B}\phi$ iff $\sigma \models \phi$,
- $\langle \sigma, \gamma \rangle \models \mathcal{G}\psi$ iff $\psi \sim_{\sigma} \gamma$,
- $\langle \sigma, \gamma \rangle \models \neg \varphi$ iff $\langle \sigma, \gamma \rangle \not\models \varphi$,
- $\langle \sigma, \gamma \rangle \models \varphi_1 \land \varphi_2$ iff $\langle \sigma, \gamma \rangle \models \varphi_1$ and $\langle \sigma, \gamma \rangle \models \varphi_2$.

A number of properties of the belief and goal modalities and the relation between these operators are listed in the following lemma. By the necessitation rule, an agent believes all tautologies ($\mathcal{B}\text{true}$). The first validity below states that the beliefs of an agent are consistent. The belief modality distributes over implication, which is expressed by the second validity. This implies that the beliefs of an agent are closed under logical consequence. The third validity is a consequence of the constraint on mental states and expresses that if an agent believes $\phi$ it does not have a goal to
achieve \( \phi \). As a consequence, an agent cannot have a goal to achieve a tautology. An agent also does not have inconsistent goals, that is, \( \neg G \text{false} \) is valid.

The goal modality is a very weak logical operator. For example, the goal modality does not distribute over implication. A counter example is provided by the goal base \( \gamma = \{ p, p \rightarrow q \} \). Even \( G(\phi \land (\phi \rightarrow \psi)) \rightarrow G\phi \) does not hold, because the agent may believe that \( \psi \) is the case even if it has a goal to achieve \( \phi \land (\phi \rightarrow \psi) \). Because of the axiom \( B\psi \rightarrow \neg G\psi \), we must have \( \neg G\psi \) in that case and we cannot conclude that \( G\psi \). From the fact that \( G\phi \) and \( G\psi \) hold, it is also not possible to conclude that \( G(\phi \land \psi) \). This reflects the fact that individual goals cannot be added to a single bigger goal; recall that two individual goals may be inconsistent (\( G\phi \land G\neg \phi \) is satisfiable) in which case taking the conjunction would lead to an inconsistent goal. In sum, most of the usual problems that many logical operators for motivational attitudes suffer from do not apply to our \( G \) operator (cf. also [12]). Finally, the conditions that allow to conclude that the agent has a (sub)goal \( \psi \) are that the agent has a goal \( \phi \) that logically entails \( \psi \) and that the agent does not believe that \( \psi \) is the case. The proof rule below then allows to conclude that \( G\psi \) holds.

**Lemma 2.8**

- \( \models \phi \Rightarrow \models B\phi \), for \( \phi \in \mathcal{L} \).
- \( \models \neg B\text{false} \).
- \( \models B(\phi \rightarrow \psi) \rightarrow (B\phi \rightarrow B\psi) \).
- \( \models B\phi \rightarrow \neg G\phi \).
- \( \models \neg G(\text{true}) \).
- \( \models \neg G(\text{false}) \).
- \( \not\models G(\phi \rightarrow \psi) \rightarrow (G\phi \rightarrow G\psi) \).
- \( \not\models G(\phi \land (\phi \rightarrow \psi)) \rightarrow G\psi \).
- \( \not\models (G\phi \land G\psi) \rightarrow G(\phi \land \psi) \).
- \[
\frac{G\phi, \neg B\psi, \models \phi \rightarrow \psi}{G\psi}
\]

Now we have defined the formal semantics of mental state formulas, we are able to formally define the selection and execution of a conditional action. The selection of an action by an agent depends on the satisfaction conditions of the mental state condition associated with the action in a conditional action. The conditions for action selection thus may express conditions on both the belief and goal base of the agent. The belief conditions associated with the action formulate preconditions on the current belief base of the agent. Only if the current beliefs of the agent satisfy these conditions, an action may be selected. A condition \( G\phi \) on the goal base is satisfied if \( \phi \) is entailed by one of the current goals of the agent (and thus, assuming the programmer did a good job, helps in bringing about one of these goals). The intuition here is that an agent is satisfied with anything bringing about at least (part of) one of its current goals. Note that a condition \( G\phi \) can only be satisfied if the agent does not already believe that \( \phi \) is the case (\( \sigma \not\models \phi \)) which prevents an agent from performing an action without any need to do so.

In the definition below, we assume that the action component \( \Pi \) of an agent \( \langle \Pi, \sigma_0, \gamma_0 \rangle \) is fixed. The execution of an action gives rise to a *computation step* formally denoted by the transition relation \( \xrightarrow{a} \) where \( a \) is the conditional action executed in the computation step. More than one computation step may be possible in a current state and the step relation \( \xrightarrow{-} \) thus denotes a possible computation step in a state. A computation step updates the current state and yields the next state of the computation. Note that because \( M \) is a partial function, a conditional action can only be successfully executed if both the condition is satisfied and the basic action is enabled.
Definition 2.9 (action selection)
Let \((\sigma, \gamma)\) be a mental state and \(b = \varphi \rightarrow do(a) \in \Pi\). Then, as a rule, we have:
If
- the mental condition \(\varphi\) holds in \((\sigma, \gamma)\), i.e. \(\langle \sigma, \gamma \rangle \models \varphi\), and
- \(a\) is enabled in \((\sigma, \gamma)\), i.e. \(M(a, \langle \sigma, \gamma \rangle)\) is defined,
then \((\sigma, \gamma) \xrightarrow{b} M(a, \langle \sigma, \gamma \rangle)\) is a possible computation step. The relation \(\xrightarrow{\cdot}\) is the smallest relation closed under this rule.

We say that a capability \(a \in \text{Cap}\) is enabled in a mental state \((\sigma, \gamma)\) in case \(M(a, \langle \sigma, \gamma \rangle)\) is defined. This definition implies that a belief update capability \(a \in B\text{Cap}\) is enabled if \(T(a, \sigma)\) is defined. A conditional action \(b\) is enabled in a mental state \((\sigma, \gamma)\) if there are \(\sigma', \gamma'\) such that \((\sigma, \gamma) \xrightarrow{b} (\sigma', \gamma')\). Note that if a capability \(a\) is not enabled, a conditional action \(\varphi \rightarrow do(a)\) is also not enabled. The special predicate \(\text{enabled}\) is introduced to denote that a capability \(a\) or conditional action \(b\) is enabled (denoted by \(\text{enabled}(a)\) respectively \(\text{enabled}(b)\)).

Definition 2.10 (semantics of enabled)
- \((\sigma, \gamma) \models \text{enabled}(a)\) iff \(M(a, \langle \sigma, \gamma \rangle)\) is defined for \(a \in \text{Cap}\),
- \((\sigma, \gamma) \models \text{enabled}(b)\) iff there are \(\sigma', \gamma'\) such that \((\sigma, \gamma) \xrightarrow{b} (\sigma', \gamma')\) for conditional actions where \(b = \varphi \rightarrow do(a)\).

The relation between the enabledness of capabilities and conditional actions is stated in the next lemma together with the fact that \(\text{drop}(\phi)\) is always enabled and a proof rule for deriving \(\text{enabled}(\text{adopt}(\phi))\).

Lemma 2.11
- \(\models \text{enabled}(\varphi \rightarrow do(a)) \leftrightarrow (\varphi \land \text{enabled}(a))\),
- \(\models \text{enabled}(\text{drop}(\phi))\),
- \(\models \text{enabled}(\text{adopt}(\phi)) \rightarrow \neg \text{B}\phi\),
- \(\not\models \neg \phi \quad \therefore \quad \neg \text{B}\phi \rightarrow \text{enabled}(\text{adopt}(\phi))\)

3 A Personal Assistant Example

In this section, we give an example to show how the programming language GOAL can be used to program agents. The example concerns a shopping agent that is able to buy books on the Internet on behalf of the user. The example provides for a simple illustration of how the programming language works. The agent in our example uses a standard procedure for buying a book. It first goes to a bookstore, in our case Amazon.com. At the web site of Amazon.com it searches for a particular book, and if the relevant page with the book details shows up, the agent puts the book in its shopping cart. In case the shopping cart of the agent contains some items, it is allowed to buy the items on behalf of the user. The idea is that the agent adopts a goal to buy a book if the user instructs it to do so.

The set of capabilities of the agent is defined by

\[ B\text{Cap} = \{\text{goto}_\text{website}(\text{site}), \text{search}(\text{book}), \text{put\_in\_shopping\_cart}(\text{book}), \text{pay\_cart}\} \]

The capability \(\text{goto}_\text{website}(\text{site})\) goes to the selected web page \(\text{site}\). In our example, relevant web pages are the home page of the user, the main page of Amazon.com, web pages with information
about books to buy, and a web page that shows the current items in the shopping cart of the agent. The capability search(book) is an action that can be selected at the main page of Amazon.com and selects the web page with information about book. The action put_in_shopping_cart(book) can be selected on the page concerning book and puts book in the cart; a new web page called ContentCart shows up showing the content of the cart. Finally, in case the cart is not empty the action pay_cart can be selected to pay for the books in the cart.

In the program text below, we assume that book is a variable referring to the specifics of the book the user wants to buy (in the example, we use variables as a means for abbreviation; variables should be thought of as being instantiated with the relevant arguments in such a way that predicates with variables reduce to propositions). The initial beliefs of the agent are that the current web page is the home page of the user, and that it is not possible to be on two different web pages at the same time. We also assume that the user has provided the agent with the goals to buy The Intentional Stance by Daniel Dennett and Intentions, Plans, and Practical Reason by Michael Bratman.

\[ \Pi = \{ \]
\[ B(\text{current\_website(\text{homepage(user)})}) \lor B(\text{current\_website(\text{ContentCart})}) \]
\[ \land G(\text{bought(book)}) \rightarrow do(\text{goto\_website(Amazon.com)}), \]
\[ B(\text{current\_website(Amazon.com)}) \land \neg B(\text{in\_cart(book)}) \]
\[ \land G(\text{bought(book)}) \rightarrow do(\text{search(book)}), \]
\[ B(\text{current\_website(book)}) \land G(\text{bought(book)}) \rightarrow do(\text{put\_in\_shopping\_cart(book)}), \]
\[ G(\text{pay\_cart}) \}
\[ \sigma_0 = \{ \text{current\_website(\text{homepage(user)})}, \]
\[ \forall s,s'(s \neq s' \land \text{current\_website(s)}) \rightarrow \neg \text{current\_website(s')}, \]
\[ \gamma_0 = \{ \text{bought(The Intentional Stance)} \land \text{bought(Intentions, Plans and Practical Reason)} \} \]

**GOAL Shopping Agent**

Some of the details of this program will be discussed in the sequel, when we prove some properties of the program. The agent basically follows the recipe for buying a book outlined above. For now, however, just note that the program is quite flexible, even though the agent more or less executes a fixed recipe for buying a book. The flexibility results from the agent’s knowledge state and the non-determinism of the program. In particular, the ordering in which the actions are performed by the agent - which book to find first, buy a book one at a time or both in the same shopping cart, etc. is not determined by the program. The scheduling of these actions thus is not fixed by the program, and might be fixed arbitrarily on a particular agent architecture used to run the program.

## 4 Temporal Logic for GOAL

On top of the language GOAL and its semantics, we now construct a temporal logic to prove properties of GOAL agents. The logic is similar to other temporal logics but its semantics is derived from the operational semantics for GOAL. Moreover, the logic incorporates the belief and goal modalities used in GOAL agents. First, we introduce the semantics for GOAL agents. Then we discuss basic action theories and in particular the use of Hoare triples for the specification of actions. These Hoare triples play an important role in the programming logic since it can be shown that temporal properties of agents can be proven by means of proving Hoare triples for actions only. Finally, the language for expressing temporal properties and its semantics is defined and the fact that certain classes of interesting temporal properties can be reduced to properties of actions, expressed by Hoare triples, is proven.

### 4.1 Semantics of GOAL Agents

The semantics of GOAL agents is derived directly from the operational semantics and the computation step relation \[ \rightarrow \] as defined in the previous section. The meaning of a GOAL agent consists
of a set of so called traces. A trace is an infinite computation sequence of consecutive mental states interleaved with the actions that are scheduled for execution in each of those mental states. The fact that a conditional action is scheduled for execution in a trace does not mean that it is also enabled in the particular state for which it has been scheduled. In case an action is scheduled but not enabled, the action is simply skipped and the resulting state is the same as the state before.

Definition 4.1 (trace)
A trace $s$ is an infinite sequence $s_0, b_0, s_1, b_1, s_2, \ldots$ such that $s_i$ is a mental state, $b_i$ is a conditional action, and for every $i$ we have: $s_i \xrightarrow{b_i} s_{i+1}$, or $b_i$ is not enabled in $s_i$ and $s_i = s_{i+1}$.

An important assumption in the semantics for GOAL is a fairness assumption. Fairness assumptions concern the fair selection of actions during the execution of a program. In our case, we make a weak fairness assumption [11]. A trace is weakly fair if it is not the case that an action is always enabled from some point in time on but is never selected for execution. This weak fairness assumption is built into the semantics by imposing a constraint on traces. By definition, a fair trace is a trace in which each of the actions is scheduled infinitely often. In a fair trace, there always will be a future time point at which an action is scheduled (considered for execution) and by this scheduling policy a fair trace implements the weak fairness assumption. However, note that the fact that an action is scheduled does not mean that the action also is enabled (and therefore, the selection of the action may result in an idle step which does not change the state).

The meaning of a GOAL agent now is defined as the set of fair traces in which the initial state is the initial mental state of the agent and each of the steps in the trace corresponds to the execution of a conditional action or an idle transition.

Definition 4.2 (meaning of a GOAL agent)
The meaning of a GOAL agent $\langle \Pi, \sigma_0, \gamma_0 \rangle$ is the set of fair traces $S$ such that for $s \in S$ we have $s_0 = \langle \sigma_0, \gamma_0 \rangle$.

4.2 Hoare Triples
The specification of basic actions provides the basis for the programming logic, and, as we will show below, is all we need to prove properties of agents. Because they play such an important role in the proof theory of GOAL, the specification of the basic agent capabilities requires special care. In the proof theory of GOAL, Hoare triples of the form $\{ \varphi \} b \{ \psi \}$, where $\varphi$ and $\psi$ are mental state formulas, are used to specify actions. The use of Hoare triples in a formal treatment of traditional assignments is well-understood [1]. Because the agent capabilities of GOAL agents are quite different from assignment actions, however, the traditional predicate transformer semantics is not applicable. GOAL agent capabilities are mental state transformers and, therefore, we require more extensive basic action theories to formally capture the effects of such actions. Hoare triples are used to specify the postconditions and the frame conditions of actions. The postconditions of an action specify the effects of an action whereas the frame conditions specify what is not changed by the action. Axioms for the predicate enabled specify the preconditions of actions.

The formal semantics of a Hoare triple for conditional actions is derived from the semantics of a GOAL agent and is defined relative to the set of traces $S_A$ associated with the GOAL agent $A$. A Hoare triple for conditional actions thus expresses a property of an agent and not just a property of an action. The semantics of the basic capabilities are assumed to be fixed, however, and are not defined relative to an agent.

Definition 4.3 (semantics of Hoare triples for basic actions)
A Hoare triple for basic capabilities $\{ \varphi \} a \{ \psi \}$ means that for all $\sigma, \gamma$

- $\langle \sigma, \gamma \rangle \models \varphi \land enabled(a) \Rightarrow M(a, \langle \sigma, \gamma \rangle) \models \psi$, and
- $\langle \sigma, \gamma \rangle \models \varphi \land \neg enabled(a) \Rightarrow \langle \sigma, \gamma \rangle \models \psi$. 
To explain this definition, note that we made a case distinction between states in which the basic action is enabled and in which it is not enabled. In case the action is enabled, the postcondition $\psi$ of the Hoare triple $\{\varphi\} a \{\psi\}$ should be evaluated in the next state resulting from executing action $a$. In case the action is not enabled, however, the postcondition should be evaluated in the same state because a failed attempt to execute action $a$ is interpreted as an idle step in which nothing changes.

Hoare triples for conditional actions are interpreted relative to the set of traces associated with the GOAL agent of which the action is a part. Below, we write $\varphi[s_i]$ to denote that a mental state formula $\varphi$ holds in state $s_i$.

**Definition 4.4 (semantics of Hoare triples for conditional actions)**
Given an agent $A$, a Hoare triple for conditional actions $\{\varphi\} b \{\psi\}$ (for $A$) means that for all traces $s \in S_A$ and $i$, we have that

$$(\varphi[s_i] \land b = b_i \in s) \Rightarrow \psi[s_{i+1}]$$

where $b_i \in s$ means that action $b_i$ is taken in state $i$ of trace $s$.

Of course, there is a relation between the execution of basic actions and that of conditional actions, and therefore there also is a relation between the two types of Hoare triples. The following lemma makes this relation precise.

**Lemma 4.5** Let $A$ be a GOAL agent and $S_A$ be the meaning of $A$. Suppose that we have $\{\varphi \land \psi\} a \{\varphi'\}$ and $S_A \models (\varphi \land \lnot \psi) \rightarrow \varphi'$. Then we also have $\{\varphi\} \psi \rightarrow \text{do}(a) \{\varphi'\}$.

**Proof:** We need to prove that $(\varphi[s_i] \land (\psi \rightarrow \text{do}(a)) = b_i \in s) \Rightarrow \varphi'[s_{i+1}]$. Therefore, assume $\varphi[s_i] \land (\psi \rightarrow \text{do}(a)) = b_i \in s$. Two cases need to be distinguished: The case that the condition $\psi$ holds in $s_i$ and the case that it does not hold in $s_i$. In the former case, because we have $\{\varphi \land \psi\} a \{\varphi'\}$ we then know that $s_{i+1} \models \varphi'$. In the latter case, the conditional action is not executed and $s_{i+1} = s_i$. From $((\varphi \land \lnot \psi) \rightarrow \varphi')[s_i], \varphi[s_i]$ and $\lnot \psi[s_i]$ it then follows that $\varphi'[s_{i+1}]$ since $\varphi'$ is a state formula. $\Box$

The definition of Hoare triples presented here formalises a total correctness property. A Hoare triple $\{\varphi\} b \{\psi\}$ ensures that if initially $\varphi$ holds, then an attempt to execute $b$ results in a successor state in which $\psi$ holds. This is different from partial correctness where no claims about the termination of actions and the existence of successor states are made.

### 4.3 Basic Action Theories

A basic action theory specifies the effects of the basic capabilities of an agent. It specifies when an action is enabled, it specifies the effects of an action and what does not change when an action is executed. Therefore, a basic action theory consists of axioms for the predicate enabled for each basic capability, Hoare triples that specify the effects of basic capabilities and Hoare triples that specify frame axioms associated with these capabilities. Since the belief update capabilities of an agent are not fixed by the language GOAL but are user-defined, the user should specify the axioms and Hoare triples for belief update capabilities. The special actions for goal updating adopt and drop are part of GOAL and a set of axioms and Hoare triples for these actions is specified below.

Because in this paper, our concern is not with the specification of basic action theories in particular, but with providing a programming framework for agents in which such specifications can be plugged in, we only provide some example specifications of the capabilities defined in the personal assistant example that we need in the proof of correctness below.

First, we specify a set of axioms for each of our basic actions that state when that action is enabled. Below, we abbreviate the book titles of the example, and write $T$ for The Intentional
Stance and I for Intentions, Plans, and Practical Reason. In the shopping agent example, we then have:

\[
\begin{align*}
\text{enabled}(\text{goto_website}(\text{site})) & \leftrightarrow \text{true}, \\
\text{enabled}(\text{search}(\text{book})) & \leftrightarrow \not B(\text{current_website}(\text{Amazon.com})), \\
\text{enabled}(\text{put_in_shopping_cart}(\text{book})) & \leftrightarrow \not B(\text{current_website}(\text{book})), \\
\text{enabled}(\text{pay_cart}) & \leftrightarrow \left( (\text{Bin_cart}(T) \lor \text{Bin_cart}(I)) \land \text{current_website}(\text{ContentCart}) \right).
\end{align*}
\]

Second, we list a number of effect axioms that specify the effects of a capability in particular situations defined by the preconditions of the Hoare triple.

- The action \text{goto_website}(\text{site}) results in moving to the relevant web page:
  \{
  \text{true} \} \text{goto_website}(\text{site}) \{ \text{Bcurrent_website}(\text{site}) \},

- At Amazon.com, searching for a book results in finding a page with relevant information about the book:
  \{ \text{Bcurrent_website}(\text{Amazon.com}) \} \text{search}(\text{book}) \{ \text{Bcurrent_website}(\text{book}) \}

- On the page with information about a particular book, selecting the action \text{put_in_shopping_cart}(\text{book}) results in the book being put in the cart; also, a new web page appears on which the contents of the cart are listed:
  \{ \text{Bcurrent_website}(\text{book}) \}
  \text{put_in_shopping_cart}(\text{book})
  \{ \text{B(in_cart}(\text{book}) \land \text{current_website}(\text{ContentCart})) \}

- In case \text{book} is in the cart, and the current web page presents a list of all the books in the cart, the action \text{pay_cart} may be selected resulting in the buying of all listed books:
  \{ \text{B(in_cart}(\text{book}) \land \text{current_website}(\text{ContentCart})) \}
  \text{pay_cart}
  \{ \neg \text{Bin_cart}(\text{book}) \land \text{B(bought}(\text{book}) \land \text{current_website}(\text{Amazon.com})) \}

Finally, we need a number of frame axioms that specify which properties are not changed by each of the capabilities of the agent. For example, both the capabilities \text{goto_website}(\text{site}) and \text{search}(\text{book}) do not change any beliefs about \text{in_cart}. Thus we have, e.g.:

\[
\begin{align*}
\{ \text{Bin_cart}(\text{book}) \} \text{goto_website}(\text{site}) \{ \text{Bin_cart}(\text{book}) \} \\
\{ \text{Bin_cart}(\text{book}) \} \text{search}(\text{book}) \{ \text{Bin_cart}(\text{book}) \}
\end{align*}
\]

It will be clear that we need more frame axioms than these two, and some of these will be specified below in the proof of the correctness of the shopping agent.

It is important to realise that the only Hoare triples that need to be specified for agent capabilities are Hoare triples that concern the effects upon the beliefs of the agent. Changes and persistence of (some) goals due to executing actions can be derived with the proof rules and axioms below that are specifically designed to reason about the effects of actions on goals.

A theory of the belief update capabilities and their effects on the beliefs of an agent must be complemented with a theory about the effects of actions upon the goals of an agent. Such a theory should capture both the effects of the default commitment strategy as well as give a formal specification of the the \text{drop} and \text{adopt} actions.

The default commitment strategy imposes a constraint on the persistence of goals. A goal persists if it is not the case that after doing a the goal is believed to be achieved. Only action \text{drop}(\phi) is allowed to overrule this constraint. Therefore, in case \text{a} \neq \text{drop}(\phi), we have that \{ G\phi \} \text{a} \{ B\phi \lor G\phi \}. This Hoare triple precisely captures the default commitment strategy and states that after executing an action the agent either believes it has achieved \phi or it still has the
goal $\phi$ if $\phi$ was a goal initially. A similar Hoare triple can be given for the persistence of the absence of a goal. Formally, we have $\{\neg G_\phi\} b \{\neg B_\phi \lor \neg G_\phi\}$. This Hoare triple states that the absence of a goal $\phi$ persists, and in case it does not persist the agent does not believe $\phi$ (anymore). The adoption of a goal may be the result of executing an adopt action, of course. However, it may also be the case that an agent believed it achieved $\phi$ but after doing $b$ no longer believes this to be the case and adopts $\phi$ as a goal again. For example, if the goal base $\gamma = \{ p \land q \}$ and the belief base $\sigma = \{ p \}$, then the agent does not have a goal to achieve $p$ because it already believes $p$ to be the case; however, in case an action changes the belief base such that $p$ is no longer is believed, the agent has a goal to achieve $p$ (again). This provides for a mechanism similar to that of maintenance goals. We do not need the Hoare triple as an axiom, however, since it is a direct consequence of the fact that $B_\phi \rightarrow \neg G_\phi$. Note that the stronger $\{\neg G_\phi\} b \{\neg G_\phi\}$ does not hold, even if $b \neq \varphi \rightarrow do(\text{adopt}(\varphi))$.

The specification of the special actions drop and adopt involves a number of frame axioms and a number of proof rules. The frame axioms capture the fact that neither of these actions has any effect on the beliefs of an agent:

- $\{ B_\phi \} \text{ adopt}(\psi) \{ B_\psi \}$, $\{ \neg B_\phi \} \text{ adopt}(\psi) \{ \neg B_\psi \}$.
- $\{ B_\phi \} \text{ drop}(\psi) \{ B_\psi \}$, $\{ \neg B_\phi \} \text{ drop}(\psi) \{ \neg B_\psi \}$.

The proof rules for the actions adopt and drop capture the effects on the goals of an agent. For each action, we list proof rules for the adoption respectively the dropping of goals, and for the persistence of goals. An agent adopts a new goal $\phi$ in case the agent does not believe $\phi$ and $\phi$ is not a contradiction.

$$
\begin{array}{c}
\phi \\
\\neg \phi
\end{array} \Rightarrow \begin{array}{c}
\{ \neg B_\phi \} \text{ adopt}(\phi) \{ G_\phi \}
\end{array}
$$

An adopt action does not remove any current goals of the agent. Any existing goals thus persist when adopt is executed. The persistence of the absence of goals is somewhat more complicated in the case of an adopt action. An adopt($\phi$) action does not add a new goal $\psi$ in case $\psi$ is not entailed by $\phi$ or $\psi$ is believed to be the case:

$$
\begin{array}{c}
\phi \\
\neg \psi \rightarrow \phi
\end{array} \Rightarrow \begin{array}{c}
\{ G_\psi \} \text{ adopt}(\phi) \{ G_\psi \}
\end{array}
$$

$$
\begin{array}{c}
\neg \psi \rightarrow \neg \phi
\end{array} \Rightarrow \begin{array}{c}
\{ \neg G_\psi \} \text{ adopt}(\phi) \{ G_\psi \}
\end{array}
$$

A drop action drop($\phi$) results in the removal of all goals that entail $\phi$. This is captured by the proof rule:

$$
\begin{array}{c}
\psi \\
\neg \psi \rightarrow \phi
\end{array} \Rightarrow \begin{array}{c}
\{ G_\psi \} \text{ drop}(\phi) \{ \neg G_\psi \}
\end{array}
$$

A drop action drop($\phi$) never results in the adoption of new goals. The absence of a goal $\psi$ thus persists when a drop action is executed. It is more difficult to formalise the persistence of a goal with respect to a drop action. Since a drop action drop($\phi$) removes goals which entail $\phi$, to conclude that a goal $\psi$ persists after executing the action, we must make sure that the goal does not depend on a goal (is a subgoal) that is removed by the drop action. In case the conjunction $\phi \land \psi$ is not a goal, we know this for certain.

$$
\begin{array}{c}
\neg \phi
\end{array} \Rightarrow \begin{array}{c}
\{ \neg G_\phi \} \text{ drop}(\psi) \{ \neg G_\phi \}
\end{array}
$$

The basic action theories for GOAL include a number of proof rules to derive Hoare triples. The Rule for Infeasible Capabilities allows to derive frame axioms for a capability in case it is not enabled in a particular situation. The Rule for Conditional Actions allows the derivation of Hoare triples for conditional actions from Hoare triples for capabilities. This rule is justified by lemma 4.5. Finally, there are three rules for combining Hoare triples and for strengthening the precondition and weakening the postcondition.
4.4 Temporal logic

On top of the Hoare triples for specifying actions, a temporal logic is used to specify and verify properties of GOAL agents. Two new operators are introduced. The proposition \text{init} states that the agent is at the beginning of execution and nothing has happened yet. The second operator \text{until} is a weak until operator. \( \varphi \text{ until } \psi \) means that \( \psi \) eventually becomes true and \( \varphi \) is true until \( \psi \) becomes true, or \( \psi \) never becomes true and \( \varphi \) remains true forever.

**Definition 4.6 (language of temporal logic \( \mathcal{L}_T \) based on \( \mathcal{L} \))**

The temporal logic language \( \mathcal{L}_T \) is inductively defined by:

- \( \text{init} \in \mathcal{L}_T \),
- \( \text{enabled}(a), \text{enabled}(\varphi \rightarrow \text{do}(a)) \in \mathcal{L}_T \) for \( a \in \text{Cap} \),
- if \( \varphi \in \mathcal{L}_T \), then \( \mathcal{B}\varphi, \mathcal{G}\varphi \in \mathcal{L}_T \),
- if \( \varphi, \psi \in \mathcal{L}_T \), then \( \neg \varphi, \varphi \land \psi \in \mathcal{L}_T \),
- if \( \varphi, \psi \in \mathcal{L}_T \), then \( \varphi \text{ until } \psi \in \mathcal{L}_T \).

A number of other well known temporal operators can be defined in terms of the operator \text{until}. The \text{always} operator \( \Box \varphi \) is an abbreviation for \( \varphi \text{ until false} \), and the \text{eventuality} operator \( \Diamond \varphi \) is defined as \( \neg \Box \neg \varphi \) as usual.

Temporal formulas are evaluated with respect to a trace \( s \) and a time point \( i \). State formulas like \( \mathcal{B}\varphi, \mathcal{G}\varphi, \text{enabled}(a) \) etc. are evaluated with respect to mental states.

**Definition 4.7 (semantics of temporal formulas)**

Let \( s \) be a trace and \( i \) be a natural number.

- \( s, i \models \text{init} \) if \( i = 0 \),
- \( s, i \models \text{enabled}(a) \) iff \( \text{enabled}(a)[s_i] \),
- \( s, i \models \text{enabled}(\varphi \rightarrow \text{do}(a)) \) iff \( \text{enabled}(\varphi \rightarrow \text{do}(a))[s_i] \),
- \( s, i \models \mathcal{B}\varphi \) iff \( \mathcal{B}\varphi[s_i] \),
- \( s, i \models \mathcal{G}\varphi \) iff \( \mathcal{G}\varphi[s_i] \),
- \( s, i \models \neg \varphi \) iff \( s, i \not\models \varphi \),
- \( s, i \models \varphi \land \psi \) iff \( s, i \models \varphi \) and \( s, i \models \psi \),
- \( s, i \models \varphi \text{ until } \psi \) iff \( \exists j \geq i(s, j \models \psi \land \forall k(i \leq k < j(s, k \models \varphi))) \) or \( \forall k \geq i(s, k \models \varphi) \).
We are particularly interested in temporal formulas that are valid with respect to the set of traces \( S_A \) associated with a GOAL agent \( A \). Temporal formulas valid with respect to \( S_A \) express properties of the agent \( A \).

**Definition 4.8** Let \( S \) be a set of traces.

- \( S \models \varphi \) iff \( \forall s \in S, i(s, i = \varphi) \).
- \( \models \varphi \) iff \( S \models \varphi \) where \( S \) is the set of all traces.

In general, two important types of temporal properties can be distinguished. Temporal properties are divided into liveness and safety properties. Liveness properties concern the progress that a program makes and express that a (good) state eventually will be reached. Safety properties, on the other hand, express that some (bad) state will never be entered. In the rest of this section, we discuss a number of specific liveness and safety properties of an agent \( A = \langle \Pi_A, \sigma_A, \gamma_A \rangle \).

We show that each of the properties that we discuss are equivalent to a set of Hoare triples. The importance of this result is that it shows that temporal properties of agents can be proven by inspection of the program text only. The fact that proofs of agent properties can be constructed by inspection of the program text means that there is no need to reason about individual traces of an agent or its operational behaviour. In general, reasoning about the program text is more economical since the number of traces associated with a program is exponential in the size of the program.

The first property we discuss concerns a safety property, and is expressed by the temporal formula \( \varphi \rightarrow (\varphi \mathbf{until} \psi) \). Properties in this context always refer to agent properties and are evaluated with respect to the set of traces associated with that agent. Therefore, we can explain the informal meaning of the property as stating that if \( \varphi \) ever becomes true, then it remains true until \( \psi \) becomes true. By definition, we write this property as \( \varphi \mathbf{unless} \psi \):

\[
\varphi \mathbf{unless} \psi \equiv \varphi \rightarrow (\varphi \mathbf{until} \psi)
\]

An important special case of an \( \mathbf{unless} \) property is \( \varphi \mathbf{unless false} \), which expresses that if \( \varphi \) ever becomes true, it will remain true. \( \varphi \mathbf{unless false} \) means that \( \varphi \) is a stable property of the agent. In case we also have \( \mathbf{init} \rightarrow \varphi \), where \( \mathbf{init} \) denotes the initial starting point of execution, \( \varphi \) is always true and it is an invariant of the program.

Now we show that \( \mathbf{unless} \) properties of an agent \( A = \langle \Pi_A, \sigma_A, \gamma_A \rangle \) are equivalent to a set of Hoare triples for basic actions in \( \Pi \). This shows that we can prove \( \mathbf{unless} \) properties by proving a set of Hoare triples. The proof relies on the fact that if we can prove that after executing any action from \( \Pi \) either \( \varphi \) persists or \( \psi \) becomes true, we can conclude that \( \varphi \mathbf{unless} \psi \).

**Theorem 4.9** Let \( A = \langle \Pi_A, \sigma_A, \gamma_A \rangle \). Then:

\[
\forall b \in \Pi_A (\{\varphi \land \neg \psi\} b \{\varphi \lor \psi\}) \text{ iff } S_A \models \varphi \mathbf{unless} \psi
\]

**Proof:** The proof from right to left is the easiest direction in the proof. Suppose \( S_A \models \varphi \mathbf{unless} \psi \) and \( s, i = \varphi \). This implies that \( s, i \models \varphi \mathbf{until} \psi \). In case we also have \( s, i = \psi \), we are done. So, assume \( s, i \models \neg \psi \) and action \( b \) is selected in the trace at state \( s_i \). From the semantics of \( \mathbf{until} \) we then know that \( \varphi \lor \psi \) holds at state \( s_{i+1} \), and we immediately obtain \( \{\varphi \land \neg \psi\} b \{\varphi \lor \psi\} \) since \( s \) and \( i \) were arbitrarily chosen trace and time point. To prove the Hoare triple for the other actions in the agent program \( A \), note that when we replace action \( b \) with another action \( c \) from \( \Pi_A \) in trace \( s \), the new trace \( s' \) is still a valid trace that is in the set \( S_A \). Because we have \( S_A \models \varphi \mathbf{unless} \psi \), we also have \( s', i \models \varphi \mathbf{unless} \psi \) and from reasoning by analogy we obtain the Hoare triple for action \( c \) (and similarly for all other actions).

We prove the left to right case by contraposition. Suppose that

\[
(\ast) \forall b \in \Pi_A (\{\varphi \land \neg \psi\} b \{\varphi \lor \psi\})
\]

14
and for some \(s \in S_A\) we have \(s, i \not\models \varphi\) unless \(\psi\). The latter fact means that we have \(s, i \models \varphi\) and \(s, i \not\models \varphi\) until \(\psi\). \(s, i \not\models \varphi\) until \(\psi\) implies that either (i) \(\psi\) is never established at some \(j \geq i\) but we do have \(\neg \varphi\) at some time point \(k > i\) or (ii) \(\psi\) is established at some time \(j > i\), but in between \(i\) and any such \(j\) it is not always the case that \(\varphi\) holds.

In the first case (i), let \(k > i\) be the smallest \(k\) such that \(s, k \not\models \varphi\). Then, we have \(s, k - 1 \models \varphi \land \neg \psi\) and \(s, k \models \neg \varphi \land \neg \psi\). In state \(s_{k - 1}\), however, either a conditional action is performed or no action is performed. From (\(\ast\)) we then derive a contradiction.

In the second case (ii), let \(k > i\) be the smallest \(k\) such that \(s, k \models \psi\). Then we know that there is a smallest \(j\) such that \(i < j < k\) and \(s, j \not\models \varphi\) (\(j \not= i\) since \(s, i \models \varphi\)). This means that we have \(s, j - 1 \models \varphi \land \neg \psi\). However, in state \(s_j\) either a conditional action is performed or no action is performed. From (\(\ast\)) we then again derive a contradiction. □

Liveness properties involve eventualities which state that some state will be reached starting from a particular situation. To express a special class of such properties, we introduce the operator \(\varphi\) ensures \(\psi\). \(\varphi\) ensures \(\psi\) informally means that \(\varphi\) guarantees the realisation of \(\psi\), and is defined as:

\[
\varphi\text{ ensures }\psi \overset{df}{=} \varphi \text{ unless } \psi \land (\varphi \rightarrow o\psi)
\]

\(\varphi\) ensures \(\psi\) thus ensures that \(\psi\) is eventually realised starting in a situation in which \(\varphi\) holds, and requires that \(\varphi\) holds until \(\psi\) is realised. For the class of ensures properties, we can show that these properties can be proven by proving a set of Hoare triples. The proof of a ensures property thus can be reduced to the proof of a set of Hoare triples.

**Theorem 4.10** Let \(A = (\Pi_A, \sigma_0, \gamma_0)\). Then:

\[
\forall b \in \Pi_A(\{\varphi \land \neg \psi\} b \{\varphi \land \psi\}) \land b \in \Pi_A(\{\varphi \land \neg \psi\} b \{\psi\})
\]

\[
\Rightarrow S_A \models \varphi\text{ ensures }\psi
\]

**Proof:** In the proof, we need the weak fairness assumption. Since \(\varphi\) ensures \(\psi\) is defined as \(\varphi\) unless \(\psi\land(\varphi \rightarrow o\psi)\), by theorem 4.9 we only need to prove that \(S_A \models \varphi \rightarrow o\psi\) given that \(\forall b \in \Pi_A(\{\varphi \land \neg \psi\} b \{\varphi \land \psi\}) \land b \in \Pi_A(\{\varphi \land \neg \psi\} b \{\psi\})\). Now suppose, to arrive at a contradiction, that for some time point \(i\) and trace \(s \in S_A\) we have: \(s, i \models \varphi \land \neg \psi\) and assume that for all later points \(j > i\) we have \(s, j \models \neg \psi\). In that case, we know that for all \(j > i\) we have \(s, j \models \varphi \land \neg \psi\) (because we may assume \(\varphi\) unless \(\psi\)). However, we also know that there is an action \(b\) that is enabled in a state in which \(\varphi \land \neg \psi\) holds and transforms this state to a state in which \(\psi\) holds. The action \(b\) thus is always enabled, but apparently never taken. This is forbidden by weak fairness, and we arrive at a contradiction. □

The implication in the other direction in theorem 4.10 does not hold. A counterexample is provided by the program:

\[
\Pi = \\
\begin{align*}
B(\neg p \land q) & \rightarrow do(ins(p)), \\
B(\neg p \land r) & \rightarrow do(ins(p)), \\
Bp & \rightarrow do(ins(\neg p \land q)), \\
Bp & \rightarrow do(ins(\neg p \land r)),
\end{align*}
\]

\[
\sigma_0 = \{p\},
\gamma_0 = \emptyset.
\]

where

\[
T(ins(p), \{\neg p \land q\}) = \{p \land q\},\quad T(ins(p), \{-p \land r\}) = \{p \land r\},
\]

\[
T(ins(\neg p \land q), \{p\}) = T(ins(\neg p \land q), \{p \land q\}) = \{-p \land q\},
\]

\[
T(ins(\neg p \land r), \{p\}) = T(ins(\neg p \land r), \{p \land r\}) = \{-p \land r\}.
\]
For this program, we have that $B \rightarrow p$ ensures $Bp$ holds, but we do not have $\{B \rightarrow p \land \neg Bp\}$ $b\{Bp\}$ for any $b \in \Pi$.

Finally, we introduce a third temporal operator ‘leads to’. The operator $\varphi \rightarrow \psi$ differs from ensures in that it does not require $\varphi$ to remain true until $\psi$ is established, and is derived from the ensures operator. $\rightarrow$ is defined as the transitive, disjunctive closure of ensures.

**Definition 4.11 (leads to operator)**
The leads to operator $\rightarrow$ is defined by:

<table>
<thead>
<tr>
<th>$\varphi$ ensures $\psi$</th>
<th>$\varphi \equiv \chi, \chi \rightarrow \psi$</th>
<th>$\varphi_1 \rightarrow \psi, \ldots, \varphi_n \rightarrow \psi$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\varphi \rightarrow \psi$</td>
<td>$\varphi_1 \rightarrow \psi, \ldots, \varphi_n \rightarrow \psi$</td>
<td></td>
</tr>
</tbody>
</table>

The meaning of the ‘leads to’ operator is captured by the following lemma. $\varphi \rightarrow \psi$ means that given $\varphi$ condition $\psi$ will eventually be realised. The proof of the lemma is an easy induction on the definition of $\rightarrow$.

**Lemma 4.12** $\varphi \rightarrow \psi \models \varphi \rightarrow \varphi \psi$.

## 5 Proving Agents Correct

In this section, we use the programming logic to prove the correctness of our example shopping agent. We do not present all the details, but provide enough details to illustrate the use of the programming logic. Before we discuss what it means that an agent program is correct and provide a proof which shows that our example agent is correct, we introduce some notation. The notation involves a number of abbreviations concerning names and propositions in the language of our example agent:

- Instead of `current_website(sitename)` we simply write `sitename`; e.g., we write `Amazon.com` and `ContentCart` instead of `current_website(Amazon.com)` and `current_website(ContentCart)`.
- As before, the book titles `The Intentional Stance` and `Intentions, Plans and Practical Reason` that the agent intends to buy are abbreviated to $T$ and $I$ respectively. These conventions can result in formulas like $B(T)$, which means that the agent is at the web page concerning the book `The Intentional Stance`.

A simple and intuitive correctness property, which is natural in this context and is applicable to our example agent, states that a GOAL agent is correct when the agent program realises the initial goals of the agent. For this subclass of correctness properties, we may consider the agent to be finished upon establishing the initial goals and in that case the agent could be terminated. Of course, it is also possible to continue the execution of such agents. This class of correctness properties can be expressed by means of temporal formulas like $G_{\phi} \rightarrow G_{\phi}$. Other correctness properties are conceivable, of course, but not all of them can be expressed easily in the temporal proof logic for GOAL.

### 5.1 Correctness Property of the Shopping Agent

From the discussion above, we conclude that the interesting property to prove for our example program is the following property:

$Bcond \land G(bought(T) \land bought(I)) \rightarrow B(bought(T) \land bought(I))$

where $Bcond$ is some condition of the initial beliefs of the agent. More specifically, $Bcond$ is defined by:

$Bcurrent_webpage(homepage(user)) \land \neg B\text{in\_cart}(T) \land \neg B\text{in\_cart}(I) \land$

$B(\forall s, s' ((s \neq s' \land current\_webpage(s)) \rightarrow \neg current\_webpage(s')))$
The correctness property states that the goal to buy the books *The Intentional Stance* and *Intentions, Plans and Practical Reason*, given some initial conditions on the beliefs of the agent, leads to buying (or believing to have bought) these books. Note that this property expresses a total correctness property. It states both that the program behaves as desired and that it will eventually reach the desired goal state. An extra reason for considering this property to express correctness of our example agent is that the goals involved once they are achieved remain true forever (they are 'stable' properties).

5.2 Invariants and Frame Axioms

To be able to prove correctness, we need a number of frame axioms. There is a close relation between frame axioms and invariants of a program. This is because frame axioms express properties that are not changed by actions, and a property that, once true, remains true whatever action is performed is a stable property. In case such a property also holds initially, the property is an invariant of the program. In our example program, there is one invariant that states that it is impossible to be at two web pages at the same time: \( \text{inv} = \forall s, s' ((s \neq s' \land \text{current\_ webpage}(s)) \rightarrow \neg \text{current\_ webpage}(s')) \).

To prove that \( \text{inv} \) is an invariant of the agent, we need frame axioms stating that when \( \text{inv} \) holds before the execution of an action it still holds after executing that action. Formally, for each \( a \in \text{Cap} \), we need: \( \{ \text{inv} \} a \{ \text{inv} \} \). These frame axioms need to be specified by the user, and for our example agent we assume that they are indeed true. By means of the Consequence Rule (strengthen the precondition of the Hoare triples for capabilities \( a \)) and the Rule for Conditional Actions (instance \( \varphi \) and \( \varphi' \) with \( \text{inv} \)), we then obtain that \( \{ \text{inv} \} b \{ \text{inv} \} \) for all \( b \in \Pi \). By theorem 4.9, we then know that \( \text{inv} \text{ unless false} \). Because we also have that initially \( \text{inv} \) holds since \( \langle \alpha_0, \gamma_0 \rangle = \text{inv} \), we may conclude that \( \text{init} \rightarrow \text{Binv \& inv unless false} \). \( \text{inv} \) thus is an invariant and holds at all times during the execution of the agent. Because of this fact, we do not mention \( \text{inv} \) explicitly anymore in the proofs below, but will freely use the property when we need it.

A second property that is stable is the property \( \text{status}(\text{book}) \):

\[
\text{status}(\text{book}) \overset{df}{=} (\text{Bin\_ cart}(\text{book}) \land \text{Bbought}(\text{book})) \lor \text{Bbought}(\text{book})
\]

The fact that \( \text{status}(\text{book}) \) is stable means that once a book is in the cart and it is a goal to buy the book, it remains in the cart and is only removed from the cart when it is bought.

The proof obligations to prove that \( \text{status}(\text{book}) \) is a stable property, i.e., to prove that \( \text{status}(\text{book}) \text{ unless false} \), consist of supplying proofs for \( \{ \text{status}(\text{book}) \} b \{ \text{status}(\text{book}) \} \) for each conditional action \( b \in \Pi \) of the shopping agent (cf. theorem 4.9). By the Rule for Conditional Actions, therefore, it is sufficient to prove for each conditional action \( \psi \rightarrow \text{do}(a) \in \Pi \) that \( \{ \text{status}(\text{book}) \land \psi \} a \{ \text{status}(\text{book}) \} \) and \( \text{status}(\text{book}) \land \neg \psi ) \rightarrow \text{status}(\text{book}) \). The latter implication is trivial. Moreover, it is clear that to prove the Hoare triples it is sufficient to prove \( \{ \text{status}(\text{book}) \} a \{ \text{status}(\text{book}) \} \) since we can strengthen the precondition by means of the Consequence Rule. The proof obligations thus reduce to proving \( \{ \text{status}(\text{book}) \} a \{ \text{status}(\text{book}) \} \) for each capability of the shopping agent.

Again, we cannot prove these Hoare triples without a number of frame axioms. Because no capability is allowed to reverse the fact that a book has been bought, for each capability, we can specify a frame axiom for the predicate \( \text{bought} \):

\[
(1) \quad \{ \text{Bbought}(\text{book}) \} a \{ \text{Bbought}(\text{book}) \}
\]

In case the book is not yet bought, selecting action \( \text{pay\_ cart} \) may change the contents of the cart and therefore we first treat the other three actions \( \text{goto\_ website}, \text{search}, \) and \( \text{put\_ in\_ shopping\_ cart} \) which are not supposed to change the contents of the cart. For each of the latter three capabilities we therefore add the frame axioms:

\[
\{ \text{Bin\_ cart}(\text{book}) \land \neg \text{Bbought}(\text{book}) \} a \{ \text{Bin\_ cart}(\text{book}) \land \neg \text{Bbought}(\text{book}) \}
\]
where \( a \neq \text{pay\_cart} \). Note that these frame axioms do not refer to goals but only refer to the beliefs of the agent, in agreement with our claim that only Hoare triples for belief updates need to be specified by the user. By using the axiom \( \text{Bought}(book) \rightarrow \neg \text{Bought}(book) \) and the Consequence Rule, however, we can conclude that:

\[
\{ \text{Bin\_cart}(book) \land \text{Bought}(book) \} \hookrightarrow \{ \text{Bin\_cart}(book) \land \neg \text{Bought}(book) \}
\]

By combining this with the axiom \( \{ \text{Bought}(book) \} \hookrightarrow \{ \text{Bought}(book) \lor \text{Bought}(book) \} \) by means of the Conjunction Rule and by rewriting the postcondition with the Consequence Rule, we then obtain

\[
\{ \text{Bin\_cart}(book) \land \text{Bought}(book) \} \hookrightarrow \{ \text{Bin\_cart}(book) \land \text{Bought}(book) \}
\]

where \( a \neq \text{pay\_cart} \). By weakening the postconditions of (1) and (2) by means of the Consequence Rule and combining the result with the Disjunction Rule, it is then possible to conclude that \( \{ \text{status}(book) \} \hookrightarrow \{ \text{status}(book) \} \) for \( a \neq \text{pay\_cart} \).

As before, in the case of capability \text{pay\_cart} we deal with each of the disjuncts of \( \text{status}(book) \) in turn. The second disjunct can be handled as before, but the first disjunct is more involved this time because \text{pay\_cart} can change both the content of the cart and the goal to buy a book if it is enabled. Note that \text{pay\_cart} only is enabled if case \text{BContentCart} holds. In case \text{BContentCart} holds and \text{pay\_cart} is enabled, from the effect axiom for \text{pay\_cart} and the Consequence Rule we obtain

\[
\{ \text{Bin\_cart}(book) \land \text{Bought}(book) \land \text{BContentCart} \} \hookrightarrow \text{pay\_cart} \{ \text{Bought}(book) \}
\]

In case \( \neg \text{BContentCart} \) holds and \text{pay\_cart} is not enabled, we use the Rule for Infeasible Capabilities to conclude that

\[
\{ \text{Bin\_cart}(book) \land \text{Bought}(book) \land \neg \text{BContentCart} \} \hookrightarrow \text{pay\_cart} \{ \text{Bought}(book) \land \neg \text{BContentCart} \}
\]

By means of the Consequence Rule and the Disjunction Rule, we then can conclude from (1), (3) and (4) that \( \{ \text{status}(book) \} \hookrightarrow \{ \text{status}(book) \} \), and we are done.

### 5.3 Proof Outline

The main proof steps to prove our agent example correct are listed next. The proof steps below consists of a number of \textit{ensures} formulas which together prove that the program reaches its goal in a finite number of steps.

1. \( \text{B\_homepage(user)} \land \neg \text{Bin\_cart}(T) \land \text{Gought}(T) \land \neg \text{Bin\_cart}(I) \land \text{Gought}(I) \) \textit{ensures} \( \text{B\_Amazon.com} \land \neg \text{Bin\_cart}(T) \land \text{Gought}(T) \land \neg \text{Bin\_cart}(I) \land \text{Gought}(I) \)
2. \( \text{B\_Amazon.com} \land \neg \text{Bin\_cart}(T) \land \text{Gought}(T) \land \neg \text{Bin\_cart}(I) \land \text{Gought}(I) \) \textit{ensures} \( \{ \text{B}(T) \land \text{Gought}(T) \land \neg \text{Bin\_cart}(I) \land \text{Gought}(I) \} \lor \{ \text{B}(I) \land \text{Gought}(I) \land \neg \text{Bin\_cart}(T) \land \text{Gought}(T) \} \)
3. \( \text{B}(T) \land \text{Gought}(T) \land \neg \text{Bin\_cart}(I) \land \text{Gought}(I) \) \textit{ensures} \( \text{Bin\_cart}(T) \land \text{Gought}(T) \land \neg \text{Bin\_cart}(I) \land \text{Gought}(I) \land \text{B\_ContentCart} \)
4. \( \text{Bin\_cart}(T) \land \text{Gought}(T) \land \neg \text{Bin\_cart}(I) \land \text{Gought}(I) \) \textit{ensures} \( \text{Amazon\_com} \land \neg \text{Bin\_cart}(I) \land \text{Gought}(I) \land \text{status}(T) \)
5. \( \text{B}(\text{Amazon\_com}) \land \neg \text{Bin\_cart}(I) \land \text{Gought}(I) \land \text{status}(T) \) \textit{ensures} \( \text{B}(I) \land \text{Gought}(I) \land \text{status}(T) \)
6. \( \text{B}(I) \land \text{Gought}(I) \land \text{status}(T) \) \textit{ensures} \( \text{Bin\_cart}(I) \land \text{Gought}(I) \land \text{B\_ContentCart} \land \text{status}(T) \)
7. \( \text{Bin\_cart}(I) \land \text{Gought}(I) \land \text{B\_ContentCart} \land \text{status}(T) \) \textit{ensures} \( \text{B\_Gought}(T) \land \text{B\_Gought}(I) \)
At step 3, the proof is split up into two subproofs, one for each of the disjuncts of the disjunct that is ensured in step 2. The proof for the other disjunct is completely analogous. By applying the rules for the ‘leads to’ operator the third to seventh step result in:

\[(a)\] \(B(T) \land G\text{bought}(T) \land \neg B\text{in\_cart}(I) \land G\text{bought}(I) \Rightarrow B\text{bought}(T) \land B\text{bought}(I)\)

\[(b)\] \(B(I) \land G\text{bought}(I) \land \neg B\text{in\_cart}(T) \land G\text{bought}(T) \Rightarrow B\text{bought}(T) \land B\text{bought}(I)\)

Combining (a) and (b) by the disjunction rule for the ‘leads to’ operator and by the transitivity of ‘leads to’ we then obtain the desired correctness result:

\[\text{Bcond} \land G(bought(T) \land bought(I)) \Rightarrow B(bought(T) \land bought(I))\]

with Bcond as defined previously.

**Step 1** We now discuss the first proof step in somewhat more detail. The remainder of the proof is left to the reader. The proof of a formula \(\varphi\) ensures \(\psi\) requires that we show that every action \(b\) in the Personal Assistant program satisfies the Hoare triple \(\{\varphi \land \neg \psi\} b \{\varphi \lor \psi\}\) and that there is at least one action \(b'\) which satisfies the Hoare triple \(\{\varphi \land \neg \psi\} b' \{\psi\}\). By inspection of the program, in our case the proof obligations turn out to be:

\[\{\text{B\text{homepage}(user)} \land \neg \text{B\text{in\_cart}(T)} \land \text{G\text{bought}(T)} \land \neg \text{B\text{in\_cart}(I)} \land \text{G\text{bought}(I)}\}\]

\[\text{B}(\text{Amazon\_com}) \land \neg \text{B(in\_cart(book))} \land \text{G(bought(book))} \rightarrow \text{do(search(book))},\]

\[\text{B(book)} \land \text{G(bought(book))} \rightarrow \text{do(put\_in\_shopping\_cart(book))},\]

\[\text{B(in\_cart(book))} \land \text{G(bought(book))} \rightarrow \text{do(pay\_cart)}\]

and

\[\{\text{B\text{homepage}(user)} \land \neg \text{B\text{in\_cart}(T)} \land \text{G\text{bought}(T)} \land \neg \text{B\text{in\_cart}(I)} \land \text{G\text{bought}(I)}\}\]

\[\text{B\text{homepage}(user)} \lor \text{ContentCart} \land \text{G(bought(book))} \rightarrow \text{do(goto\_website(Amazon\_com))}\]

\[\{\text{Amazon\_com} \land \neg \text{B\text{in\_cart}(T)} \land \text{G\text{bought}(T)} \land \neg \text{B\text{in\_cart}(I)} \land \text{G\text{bought}(I)}\}\]

The proofs of the first three Hoare triples are derived by using the Rule for Conditional Actions. The key point is noticing that each of the conditions of the conditional actions involved refers to a webpage different from the webpage homepage(user) referred to in the preconditions of the Hoare triple. The proof thus consists of using the fact that initially Bhomepage(user) and the invariant inv to derive an inconsistency which immediately yield the desired Hoare triples by means of the Rule for Conditional Actions.

To prove the Hoare triple for
\[\text{B(homepage(user)} \lor \text{ContentCart} \land \text{G(\text{bought(book))} \rightarrow \text{do(goto\_website(Amazon\_com)\}}\)

we use the effect axiom (*) for goto\_website and the frame axiom (**):

\[\{\text{B\text{ homepage(user)}}\}\]

\[\text{goto\_website(Amazon\_com)}\]

\[\{\text{B\text{Amazon\_com))}\]

and

\[\{\neg \text{B\text{in\_cart(book)}} \land \neg \text{B\text{bought(book)}}\]\n
\[\{\text{B\text{Amazon\_com)}\]

\[\{\neg \text{B\text{in\_cart(book)}} \land \neg \text{B\text{bought(book)}}\]

By using the axiom \(\text{G(bought(book))} \rightarrow \text{goto\_website(Amazon\_com)\}, the Conjunction Rule and the Rule for Conditional Actions it is then not difficult to obtain the desired conclusion.
6 Possible Extensions of GOAL

Although the basic features of the language GOAL are quite simple, the programming language GOAL is already quite powerful and can be used to program real agents. In particular, GOAL only allows the use of basic actions. There are, however, several strategies to deal with this restriction. First of all, if a GOAL agent is proven correct, any scheduling of the basic actions that is weakly fair can be used to execute the agent. More specifically, an interesting possibility is to define a mapping from GOAL agents to a particular agent architecture (cf. also [2]). As long as the agent architecture implements a weakly fair scheduling policy, concerns like the efficiency or flexibility may determine the specific mapping that is most useful with respect to available architectures.

A second strategy concerns the grain of atomicity that is required. If a coarse-grained atomicity of basic actions is feasible for an application, one might consider taking complex plans as atomic actions and instantiate the basic actions in GOAL with these plans (however, termination of these complex plans should be guaranteed). Finally, in future research the extension of GOAL with a richer notion of action structure like for example plans could be explored. This would make the programming language more practical. The addition of such a richer notion, however, is not straightforward. At a minimum, more bookkeeping seems to be required to keep track of the goals that an agent already has chosen a plan for and which it is currently executing. This bookkeeping is needed, for example, to prevent the selection of more than one plan to achieve the same goal. Note that this problem was dealt with in GOAL by the immediate and complete execution of a selected action. It is therefore not yet clear how to give a semantics to a variant of GOAL extended with complex plans. The ideal, however, would be to combine the language GOAL which includes declarative goals with our previous work on the agent programming language 3APL which includes planning features into a single new programming framework.

Apart from introducing more complex action structures, it would also be particularly interesting to extend GOAL with high-level communication primitives. Because both declarative knowledge as well as declarative goals are present in GOAL, communication primitives could be defined in the spirit of speech act theory [16]. The semantics of, for example, a request primitive could then be formally defined in terms of the knowledge and goals of an agent. Moreover, such a semantics would have a computational interpretation because both beliefs and goals have a computational interpretation in our framework.

Finally, there are a number of interesting extensions and problems to be investigated in relation to the programming logic. For example, it would be interesting to develop a semantics for the programming logic for GOAL that would allow the nesting of the belief and goal operators. In the programming logic, we cannot yet nest knowledge modalities which would allow an agent to reason about its own knowledge or that of other agents. Moreover, it is not yet possible to combine the belief and goal modalities. It is therefore not possible for an agent to have a goal to obtain knowledge, nor can an agent have explicit rather than implicit knowledge about its own goals or those of other agents. So far, the use of the B and G operators in GOAL is, first of all, to distinguish between beliefs and goals. Secondly, it enables an agent to express that it does not have a particular belief or goal (consider the difference between $\neg B \phi$ and $B \neg \phi$). Another important research issue concerns an extension of the programming framework to incorporate first order languages and extend the programming logic with quantifiers. Finally, more work needs to be done to investigate and classify useful correctness properties of agents. In conclusion, whereas the main aim may be a unified programming framework which includes both declarative goals and planning features, there is still a lot of work to be done to explore and manage the complexities of the language GOAL itself.

7 Conclusion

Although a programming language dedicated to agent programming is not the only viable approach to building agents, we believe it is one of the more practical approaches for developing agents. Several other approaches to the design and implementation of agents have been proposed. One
such approach promotes the use of agent logics for the specification of agent systems and aims at a further refinement of such specifications by means of an associated design methodology for the particular logic in use to implementations which meet this specification in, for example, an object-oriented programming language like Java. In this approach, there is no requirement on the existence of a natural mapping relating the end result of this development process - a Java implementation - and the formal specification in the logic. It is, however, not very clear how to implement these ideas for agent logics incorporating both informational and motivational attitudes and some researchers seem to have concluded from this that the notion of a motivational attitude (like a goal) is less useful than hoped for. Still another approach consists in the construction of agent architectures which 'implement' the different mental concepts. Such an architecture provides a template which can be instantiated with the relevant beliefs, goals, etc. Although this second approach is more practical than the first one, our main problem with this approach is that the architectures proposed so far tend to be quite complex. As a consequence, it is quite difficult to understand what behaviour an architecture that is instantiated will generate.

For these reasons, our own research concerning intelligent agents has focused on the programming language 3APL which supports the construction of intelligent agents, and reflects in a natural way the intentional concepts used to design agents (in contrast with the approach discussed above which promotes the use of logic, but at the same time suggests that such an intermediate level is not required).

Nevertheless, in previous work the incorporation of declarative goals in agent programming frameworks has, to our knowledge, not been established. It has been our aim in this paper to show that it is feasible to incorporate declarative goals into a programming framework (and there is no need to dismiss the concept). Moreover, our semantics is a computational semantics and it is rather straightforward to implement the language, although this may require some restrictions on the logical reasoning involved on the part of GOAL agents.

In this paper, we provided a complete programming theory. The theory includes a concrete proposal for a programming language and a formal, operational semantics for this language as well as a corresponding proof theory based on temporal logic. The logic enables reasoning about the dynamics of agents and about the beliefs and goals of the agent at any particular state during its execution. The semantics of the logic is provided by the GOAL program semantics which guarantees that properties proven in the logic are properties of a GOAL program. By providing such a formal relation between an agent programming language and an agent logic, we were able to bridge the gap between theory and practice. Moreover, a lot of work has already been done in providing practical verification tools for temporal proof theories [20].

Finally, our work shows that the (re)use of ideas and techniques from concurrent programming can be very fruitful. In particular, we have used many ideas from concurrent programming and temporal logics for programs in developing GOAL. It remains fruitful to explore and exploit ideas and techniques from these areas.

References


21


