GOAL as a Planning Formalism

Koen V. Hindriks[1] and Tijmen Roberti[1]

Delft University of Technology, Delft, The Netherlands

Abstract. It has been observed that there are interesting relations between planning and agent programming. This is not surprising as agent programming was partially motivated by the lack of planners that are able to operate in dynamic, complex environments. Vice versa it has also been observed, however, that agent programming languages typically lack planning capabilities. We show in this paper that the agent programming language GOAL is not only a programming language but can actually be used as a planning formalism as well. This opens up many possibilities for various approaches to mix execution and planning in agent-oriented programming. Moreover, by using the recently introduced *temporal* GOAL we are able to include not only the stratified axioms and ADL that are part of PDDL but also plan constraints.

1 Introduction

It has been argued in e.g. [3, 13, 14] that the combination and integration of planners into agent programming languages has many benefits. By combining the strengths of planners with the flexibility of agent programs it may be possible to handle dynamic domains more effectively, and, moreover, it becomes possible to exploit the advances of automated planning in agent programming.

Existing work on formally relating programming languages and planning formalisms, as far as we know, has mainly focussed on the language Golog. For example, in [13] the relative expressiveness of ADL [12] and Golog is investigated and a maximal fragment of so-called *basic action theories* used in Golog is identified that is expressively equivalent to ADL. We are not aware, however, of any work that formally relates agent programming languages that are based on BDI concepts such as beliefs and goals to planning from first principles.

The main contribution of this paper is to formally show that the agent programming language GOAL [10] can be used as a planning formalism. GOAL agents are BDI agents that derive their choice of action from their beliefs and goals. More specifically, we show that a fragment of PDDL including *axioms*, *ADL* and *(temporal) plan constraints* [8] can be compiled into GOAL. This result is important because it provides a clear interface between agent programs and planners. Such an interface can be used to call planners from agent programs and, vice versa, import results from research on planning into agent programming languages. It also shows that GOAL agents can solve solvable PDDL problems for a PDDL fragment including conditional effects and plan constraints. Moreover,

it clarifies the overlap and difference in concepts present in agent programming languages and planning formalisms.

A planning formalism uses an underlying knowledge representation (e.g. first-order logic in PDDL). We introduce the notion of a GOAL *framework* to separate the underlying knowledge representation used by GOAL agents from the features of the language GOAL itself. This allows us to clearly separate the features of the GOAL language that have been used to obtain our results from those provided by the knowledge representation language. We believe that this approach may also be helpful in clarifying how our results may be applied to other agent programming languages that have declarative beliefs and goals.

The paper is organized as follows. In Section 2 we introduce the syntax and semantics of the PDDL 3.0 Level 1 fragment (without preferences). Section 3 introduces the agent programming language GOAL. In Section 4 we show how PDDL problems can be compiled into GOAL. Section 5 concludes the paper.

2 PDDL Fragment: Axioms + ADL + Plan Constraints

The fragment of PDDL that we consider here is PDDL 3.0 Level 1, including axioms, ADL, and plan constraints, but excluding preferences. We will refer to this fragment as $PDDL_{Ax+CE+PC}$.

We assume that a first-order function-free language $\mathcal{L}_0$ is given that is built from a set of predicates $\mathcal{P}$, variables $\mathcal{V}$, and constants $\mathcal{C}$ in the usual way and includes equality $=$. Note that constants are allowed although $\mathcal{L}_0$ is otherwise assumed to be function-free. The set of predicates is divided into two disjunct sets of so-called *basic* predicates $\mathcal{B}$ and *derived* predicates $\mathcal{D}$. PDDL also allows typed variables but for reasons of space we do not discuss this feature. Formulae $\phi \in \mathcal{L}_0$ are also called *state formulae*, and we write $\phi[\boldsymbol{x}]$ to indicate that all free variables of ϕ occur in the vector $\boldsymbol{x}$. State formulae are used in PDDL to define the goal state $\mathcal{G}$ and as precondition of action operators. It is common to also call state formulae *goal descriptions*. We define PDDL axiom sets as in [15].

Definition 1. *(PDDL Axiom)*
A PDDL axiom *is a closed formula of the form* $\forall \boldsymbol{x}.\phi \rightarrow d(\boldsymbol{x})$*, where* $d \in \mathcal{D}$ *and* $\phi \in \mathcal{L}_0$ *(whose free variables are in the vector* $\boldsymbol{x}$*).*

PDDL axioms in this context are best thought of as definitions of derived predicates d (cf. the completion of axioms in the sense of [1]). A closed world semantics is used (see below) where $d(\boldsymbol{x})$ is false whenever it cannot be derived as true. In order to define PDDL axiom sets we use the notion of a *Negation Normal Form (NNF)*. A formula $\phi \in \mathcal{L}_0$ is in negation normal form iff negation occurs directly in front of atoms.

Definition 2. *(PDDL Axiom Set)*
A PDDL Axiom Set *is a set of PDDL axioms that is stratified. A PDDL axiom set* $\mathcal{A}$ *is stratified iff there exists a partition of the set of derived predicates* $\mathcal{D}$ *into (non-empty) subsets* $\{\mathcal{D}_i, 1 \leq i \leq n\}$ *such that for every* $d_i \in \mathcal{D}_i$ *and every axiom* $\forall \boldsymbol{x}.\phi \rightarrow d_i(\boldsymbol{x}) \in \mathcal{A}$ *we have that:*

1. *if* $d_j \in \mathcal{D}_j$ *occurs positively in an NNF of* ϕ*, then* $j \leq i$
2. *if* $d_j \in \mathcal{D}_j$ *occurs negated in an NNF of* ϕ*, then* $j < i$

The semantics of $\mathcal{L}_0$ is defined relative to a state and axiom set [15]. A PDDL *state* S is set of ground positive literals from $\mathcal{B}$. The closed world assumption applies, so any ground positive literal not in S is assumed to be false. Axioms, however, need to be treated separately and we first assume the consequences of axioms $\mathcal{A}$ are given by a set D of atoms of the form $d(\boldsymbol{x})$ with $d \in \mathcal{D}$.

Definition 3. *(Semantics of* $\mathcal{L}_0$*)*
Let $\mathcal{C}$ *be the constants of* $\mathcal{L}_0$*,* S *be a PDDL state,* D *a set of atoms* $d(\boldsymbol{x})$ *with* $d \in \mathcal{D}$*, and* $\mathcal{A}$ *an axiom set. We only provide the most important clauses:*

$$\begin{array}{ll} \langle S, D\rangle \models p(\boldsymbol{t}) & \textit{iff } p(\boldsymbol{t}) \in S \cup D \\ \langle S, D\rangle \models \neg\phi & \textit{iff } \langle S, D\rangle \not\models \phi \\ \langle S, D\rangle \models \forall x.\phi[x] & \textit{iff } \langle S, D\rangle \models \phi[c] \textit{ for all } c \in \mathcal{C} \end{array}$$

Intuitively, we can derive $d(\boldsymbol{t})$ using axiom $a = \forall \boldsymbol{x}.\phi \rightarrow d(\boldsymbol{x})$ if we have $\langle S, D\rangle \models \phi[\boldsymbol{t}]$, and add $d(\boldsymbol{t})$ to D if not already present; we write $[\![a]\!](S, D) = \{d(\boldsymbol{t}) \mid \langle S, D\rangle \models \phi[\boldsymbol{t}], \boldsymbol{t} \text{ is ground}\}$ to denote these consequences. Then the set of consequences of an axiom set $\mathcal{A}$ can be computed as follows, assuming that we have a stratification $\{A_i, 1 \leq i \leq n\}$ of $\mathcal{A}$ (cf. [15]): define $[\![\mathcal{A}]\!]_0(S) = \emptyset$, and, for $1 \leq i \leq n$, define

$$[\![\mathcal{A}]\!]_i(S) = \bigcap \left\{ D \mid \bigcup_{a \in A_i} [\![a]\!](S, D) \cup [\![A]\!]_{i-1}(S) \subseteq D \right\}$$

Finally, $S \models_{\mathcal{A}} \phi$ is defined as $\langle S, [\![\mathcal{A}]\!](S)\rangle \models \phi$.

Action operators in the ADL fragment of PDDL specify the preconditions and (conditional) effects of actions. Performing an action changes the state S.

Definition 4. *(Action Operators)*
A PDDL action operator α *is a triple* $\langle \boldsymbol{x}, \pi_\alpha, \epsilon_\alpha\rangle$ *where* $\boldsymbol{x}$ *are the action's parameters,* $\pi_\alpha \in \mathcal{L}_0$ *defines when the action can be (successfully) executed, and* ϵ_α *is set of conditions of the form* $\phi \Rightarrow \delta$ *with* $\phi \in \mathcal{L}_0$ *and* δ *a set of literals. All free variables in* π_α *and* ϵ_α *must also occur in* $\boldsymbol{x}$*.*

The effect of an action α on a state S can be derived by computing the positive effects Eff^+_{ADL} and negative effects Eff^-_{ADL}. Given that $pos(\delta)$ and $neg(\delta)$ return respectively the positive and negative literals in δ, these are defined by:

$$\mathit{Eff}^+_{ADL} = \{l \in pos(\delta) \mid S \models_{\mathcal{A}} \phi, \phi \Rightarrow \delta \in \epsilon_\alpha\}$$
$$\mathit{Eff}^-_{ADL} = \{l \in neg(\delta) \mid S \models_{\mathcal{A}} \phi, \phi \Rightarrow \delta \in \epsilon_\alpha\}$$

If the precondition of a ground operator α with effect ϵ_α holds in the current state, i.e. $S \models_{\mathcal{A}} \pi_\alpha$, then the successor state $\gamma(S, \alpha)$ is defined by:

$$\gamma(S, \alpha) = (S \setminus \mathit{Eff}^-_{ADL}) \cup \mathit{Eff}^+_{ADL}$$

A *plan* π, which is a sequence of action operators $\langle \alpha_0, \ldots, \alpha_{n-1} \rangle$, generates a sequence of states: $\langle S_0, \ldots, S_n \rangle = \langle S_0, \gamma(S_0, a_0), \gamma(S_1, a_1), \ldots, \gamma(S_{n-1}, a_{n-1}) \rangle$. Such a sequence is also called a *state trajectory.*

The language $\mathcal{L}_0$, axioms, and action operators are combined into a *planning domain definition* Δ, which is formally defined by $\Delta = \langle \mathcal{L}_0, \mathcal{A}, \mathcal{O} \rangle$, where $\mathcal{L}_0$ is a first-order function-free language based on $\mathcal{B}$, $\mathcal{D}$, $\mathcal{V}$, and $\mathcal{C}$, $\mathcal{A}$ is a stratified axiom set, and $\mathcal{O}$ is a set of action operators.

The final part of the PDDL fragment that we consider here are plan constraints (we do not discuss preferences). Plan contraints are like goals but apply to the state trajectory of a plan instead of only to the final state. A limited number of temporal modalities to express constraints are available, and we omit modalities that require explicit reference to time such as the `within` modality.

Definition 5. *(Plan Constraint)*
A plan constraint Φ *is defined as follows:* $\Phi = X\phi \mid \Phi \wedge \Phi \mid \forall \boldsymbol{x}.\Phi[\boldsymbol{x}]$*, where* X *is one of the modalities* `at end`*,* `always`*,* `sometime`*,* `at-most-once`*,* `sometime-after`*,* `sometime-before` *and* $\phi \in \mathcal{L}_0$ *a state formula.*

In contrast with standard linear temporal logic, plan constraints are evaluated relative to a *finite* state trajectory $\langle S_0, \ldots, S_n \rangle$ generated by a plan, and, as before, an axiom set $\mathcal{A}$.

Definition 6. *(Semantics of Plan Constraints)*
The semantics of temporal constraints is defined by the following clauses:

$$
\begin{array}{lll}
\langle S_0, \ldots, S_n \rangle \models_{\mathcal{A}} \texttt{at end}\ \phi & \textit{iff} & S_n \models_{\mathcal{A}} \phi \\
\langle S_0, \ldots, S_n \rangle \models_{\mathcal{A}} \texttt{always}\ \phi & \textit{iff} & \forall i : 0 \leq i \leq n : S_i \models_{\mathcal{A}} \phi \\
\langle S_0, \ldots, S_n \rangle \models_{\mathcal{A}} \texttt{sometime}\ \phi & \textit{iff} & \exists i : 0 \leq i \leq n : S_i \models_{\mathcal{A}} \phi \\
\langle S_0, \ldots, S_n \rangle \models_{\mathcal{A}} \texttt{at-most-once}\ \phi & \textit{iff} & \exists i : 0 \leq i \leq n : S_i \models_{\mathcal{A}} \phi \Rightarrow \\
 & & \neg \exists j, k : i < j < k \leq n : S_j \models_{\mathcal{A}} \neg\phi \,\&\, S_k \models_{\mathcal{A}} \phi \\
\langle S_0, \ldots, S_n \rangle \models_{\mathcal{A}} \texttt{sometime-after}\ \phi\ \psi & \textit{iff} & \exists i : 0 \leq i \leq n : S_i \models_{\mathcal{A}} \phi \Rightarrow \\
 & & \exists j : i < j \leq n : S_j \models_{\mathcal{A}} \psi \\
\langle S_0, \ldots, S_n \rangle \models_{\mathcal{A}} \texttt{sometime-before}\ \phi\ \psi & \textit{iff} & \exists i : 0 \leq i \leq n : S_i \models_{\mathcal{A}} \phi \Rightarrow \\
 & & \exists j : 0 \leq j < i : S_j \models_{\mathcal{A}} \psi
\end{array}
$$

We now have all the ingredients that are needed to define a *PDDL problem.* A PDDL problem extends a domain with more specific information regarding the initial state (which literals are true and false initially) and the goal the plan should achieve. Additionally, plan constraints may be provided that must also be satisfied by a plan.

Definition 7. *(PDDL Problem)*
A $PDDL_{Ax+CE+PC}$ planning problem Π *is a tuple* $\langle \Delta, \mathcal{C}, \mathcal{I}, \mathcal{G}, \mathcal{PC} \rangle$*, where* Δ *is a domain definition,* $\mathcal{C}$ *is a set of constants,* $\mathcal{I}$ *is the initial state,* $\mathcal{G} \in \mathcal{L}_0$ *is a closed formula called the* goal description*, and* $\mathcal{PC}$ *is a set of plan constraints.*

A plan π is said to be a *solution* for a planning problem Π iff the plan can be executed, the associated plan constraints are satisfied by the state trajectory of the plan and the goal description $\mathcal{G}$ of the problem is satisfied by the final state of that trajectory. Since $\mathcal{G}$ must be evaluated at the end of the trajectory we also sometimes proceed as if $\mathcal{G}$ is of the form `at end` ϕ.

3 The Agent Programming Language GOAL

The agent programming language GOAL is a language for programming rational agents [10, 4]. It provides constructs for specifying the *beliefs* and *goals* of agents and for programming a *strategy for action selection*. GOAL agents derive their choice of action from their beliefs and goals. GOAL defines a programming *framework* rather than a concrete programming *language* because GOAL does not commit to any particular knowledge representation and may be combined with various knowledge representation languages such as Prolog, ASP, OWL, etc. The current implementation of GOAL is based on Prolog.

3.1 GOAL Framework

We assume that some *knowledge representation technology* is available that agents use to represent, reason, and update their beliefs and goals.

Definition 8. *(Knowledge Representation Technology)*
A knowledge representation technology (KRT) *is a triple* $\langle \mathcal{L}, \models, \oplus \rangle$ *with* $\mathcal{L}$ *a* language *to represent an agent's beliefs and goals,* $\models \subseteq 2^{\mathcal{L}} \times \mathcal{L}$ *a* consequence relation *for* $\mathcal{L}$*, and* $\oplus : 2^{\mathcal{L}} \times \mathcal{L} \mapsto 2^{\mathcal{L}}$ *an* update operator *that defines how a set of formulae is updated with a given formula. We assume that falsity* $\bot \in \mathcal{L}$*.*

A KRT $\langle \mathcal{L}, \models, \oplus \rangle$is a *plugin* for a GOAL framework. A second plugin component of a GOAL framework is a *set of actions Act* that GOAL agents may perform, typically dependent on the environment of the agents. A GOAL framework is abstractly defined first and then each component of a framework is explained.

Definition 9. *(*GOAL *Framework)*
A GOAL framework *based on a KRT* $\langle \mathcal{L}, \models, \oplus \rangle$*is a tuple* $\langle \Psi, \mathcal{L}_\Psi, \models_\Psi, \mathcal{M} \rangle$*where:*

- $\Psi \subseteq \mathcal{L} \times \mathcal{L} \times \mathcal{L}$ *is the set of possible* mental states *of* GOAL *agents,*
- $\mathcal{L}_\Psi$ *is a language of* mental state conditions*,*
- $\models_\Psi \subseteq \Psi \times \mathcal{L}_\Psi$ *defines the truth conditions of mental state conditions, and*
- $\mathcal{M} : \Psi \times Act \rightharpoonup \Psi$ *is a* mental state transformer*.*

A *mental state* $m \in \Psi$ is a triple $m = \langle K, \Sigma, \Gamma \rangle$ with $K, \Sigma, \Gamma \subseteq \mathcal{L}$ which consists of a *knowledge base* K, *belief base* Σ, and *goal base* Γ. The knowledge base of a GOAL agent consists of *static conceptual or domain knowledge* that does not change. This means that performing an action does not modify a knowledge base and applying the mental state transformer $\mathcal{M}(\langle K, \Sigma, \Gamma \rangle, \alpha) = \langle K', \Sigma', \Gamma' \rangle$ is constrained such that $K = K'$. The belief base consists of the beliefs of an agent that may change due to actions that are performed. Assuming that φ represents the effects of performing action α, the belief update operator $\oplus$ is used to compute the new set of beliefs and applying the mental state transformer $\mathcal{M}(\langle K, \Sigma, \Gamma \rangle, \alpha) = \langle K', \Sigma', \Gamma' \rangle$ is constrained such that $\Sigma' = \Sigma \oplus \varphi$. The goal base consists of the goals of an agent, e.g. to have one block on top of another (sometime in the future). Typically, additional *rationality constraints* are imposed on mental states, such as that K, Σ, and Γ are consistent (i.e. $\bot$ does not

follow from any of these sets). The language $\mathcal{L}_\Psi$ of *mental state conditions* is used by GOAL agents to inspect their mental state. It consists of *mental atoms* $\mathbf{B}(\varphi)$, to inspect an agent's beliefs, and $\mathbf{G}(\varphi)$, to inspect an agent's goals, where $\varphi \in \mathcal{L}$, and combinations of such atoms by means of Boolean operators. It is not allowed to nest the operators $\mathbf{B}$ and $\mathbf{G}$. The semantics $\models_\Psi$ of mental state conditions is derived from the consequence relation $\models$ provided by the KRT, and is defined as follows (the Boolean operators are defined as usual):

$$\begin{array}{l} \langle K, \Sigma, \Gamma \rangle \models_\Psi \mathbf{B}(\varphi) \;\text{ iff }\; K \cup \Sigma \models \varphi \\ \langle K, \Sigma, \Gamma \rangle \models_\Psi \mathbf{G}(\varphi) \;\text{ iff }\; K \cup \Gamma \models \varphi \end{array}$$

This definition also clarifies the distinct role of knowledge and beliefs. Conceptual knowledge may be used *in combination both with beliefs and goals*, which allows e.g. an agent to conclude that it wants to put block a *above* block c if it wants block a on top of b on top of c using a rule that defines the concept *above*.

The actions *Act* that GOAL agents may perform are specified as STRIPS-style triples $\langle \alpha(\boldsymbol{x}), \varphi, \varphi' \rangle$ where α is the name of the action - including parameters $\boldsymbol{x}$, $\varphi \in \mathcal{L}$ is the action's *precondition*, and $\varphi' \in \mathcal{L}$ is the action's *postcondition*. Multiple action specifications for the same action α can be provided, allowing for non-deterministic actions. An action is said to be *enabled* when its precondition holds. Agents do not have direct access to their environment and have to inspect their mental state (beliefs) to verify that an action is enabled. A GOAL agent makes a choice which action it will perform from the possibly multiple actions that are enabled by a *rule-based action selection mechanism* that uses so-called *action rules*. Action rules are of the form **if** ψ **then** α and define a strategy or policy for action selection of an agent. Here, ψ is a mental state condition that specifies when action α may be selected. If ψ follows from the agent's current mental state, we say that α is *applicable*. Finally, if an action is both applicable and enabled, it is said to be an *option*, one of which is non-deterministically choosen by an agent for execution.

GOAL agents thus maintain a mental state and derive their choice of action from their beliefs and goals. A GOAL agent consists of the *initial beliefs and goals*, *specifies the preconditions and effects of the actions* available to the agent, and contains a set of *action rules* to select actions for execution at runtime. Like the knowledge base, the action specifications and action rules are static.

Definition 10. *(*GOAL *Agent)*
A GOAL agent *is a tuple* $\langle K, \Sigma, \Gamma, R, A \rangle$ *where* $\langle K, \Sigma, \Gamma \rangle$ *is a mental state,* R *is a set of action rules* **if** ψ **then** α*, and* A *is a set of action specifications.*

Given the definition of a GOAL agent and the semantics of mental state conditions by $\models_\Psi$, we can define the notion of a *computation step* in which a GOAL agent performs an action.

Definition 11. *(Computation Steps)*
Let $Agt = \langle K, \Sigma, \Gamma, R, A \rangle$ *be a* GOAL *agent with a mental state* $m = \langle K, \Sigma, \Gamma \rangle$.

Then the set of possible computation steps *that Agt can perform from* $\langle K, \Sigma, \Gamma \rangle$ *is denoted by* $\longrightarrow$ *and defined by:*

$$\frac{\textbf{if } \psi \textbf{ then } \alpha \in R, \langle \alpha, \varphi, \varphi' \rangle \in A, m \models_{\Psi} \psi \wedge \mathbf{B}(\varphi)}{m \xrightarrow{\alpha} \mathcal{M}(m, \alpha)}$$

This semantics allows for non-determinism in two ways. First, if multiple actions are options, one of these actions is non-deterministically choosen. Second, if one and the same action has multiple action specifications that can be executed simultaneously, one of these specifications is choosen non-deterministically. The latter allows for non-deterministic actions such as throwing a dice.

The action semantics of GOAL induces a set of possible *computations*. A computation is defined as an infinite sequence of mental states m_i and actions α_i, such that mental state m_{i+1} is obtained from m_i by applying the transition rule of Definition 11 with action α_i. Although computations are infinite, intuitively, the actions of a finite prefix of a computation that achieve the agent's goals may be viewed as a plan that a planner may return to achieve these goals. As GOAL agents are non-deterministic, the semantics of a GOAL agent is defined as a *set* of possible computations that start in the agent's initial mental state. A GOAL agent thus may be viewed as defining a *plan search space*; below, we make this statement precise and formally show GOAL can be used as a planning formalism.

Definition 12. *(Run, Meaning of a* GOAL *Agent)*
A run or computation r *is an infinite sequence* $m_0, \alpha_0, m_1, \alpha_1, \ldots$ *of mental states* m_i *and actions* α_i *such that* $m_i \xrightarrow{\alpha_i} m_{i+1}$*, or for all* α*:* $m_i \not\xrightarrow{\alpha}$ *and* $m_j = m_i$ *for all* $j > i$ *and* $\alpha_j = \textbf{skip}$ *for all* $j \geq i$*.*

We write r_i^m *to denote the mental state at point* i *in* r *and* r_i^a *to denote the action performed at point* i *in* r*. The meaning* $\mathcal{R}_{Agt}$ *of a* GOAL *agent named Agt with initial mental state* m_0 *is the set of all runs starting in that state.*

3.2 Temporal GOAL

The instantiation of a GOAL framework with linear temporal logic as KRT is called *temporal* GOAL. Temporal GOAL has been introduced in [10] but here we use a first-order variant and show how planning problems with temporal planning constraints can be embedded in GOAL. The KRT plugin of temporal GOAL is $\langle \mathcal{L}_{LTL}, \models_{LTL}, \oplus \rangle$ where $\mathcal{L}_{LTL}$ is a first-order linear temporal language and $\models_{LTL}$ is the usual consequence relation associated with $\mathcal{L}_{LTL}$ [7]. The standard language of linear temporal logic is extended with two special predicates $\mathbf{do}(\alpha)$ with $\alpha \in Act$ and *fail*, where $\mathbf{do}(\alpha)$ means that action α is performed and *fail* indicates a failure to achieve a goal. Formally, $\mathcal{L}_{LTL}$ is defined as an extension of $\mathcal{L}_0$ with temporal operators to facilitate the compilation of PDDL to GOAL.

Definition 13. *(Linear Temporal Logic)*
The language $\mathcal{L}_{LTL}$*, with typical element* χ*, is defined by:*

$$\chi ::= \top \mid \mathit{fail} \mid (\phi \in \mathcal{L}_0) \mid \mathbf{do}(\alpha \in Act) \mid \neg\chi \mid \chi \wedge \chi \mid \forall(x \in \mathcal{V}).\chi \mid \bigcirc\chi \mid \chi \textbf{ until } \chi$$

$\Diamond\chi$ and $\Box\chi$ are the usual abbreviations and we use χ **before** χ' as abbreviation for $\neg(\neg\chi$ **until** $\chi') \wedge \Diamond\chi'$. Temporal GOAL uses an encoding of action specifications or planning operators into $\mathcal{L}_{LTL}$ as in [5, 10, 11]. This allows us to incorporate a *declarative* encoding of action preconditions and effects in the *knowledge base* of a GOAL agent. Briefly, preconditions π_α are mapped onto *precondition axioms* of the form $\Box(\mathbf{do}(\alpha) \rightarrow \pi_\alpha)$ expressing that action α may be performed only if its precondition π_α holds. Effects are represented by a temporal encoding of successor state axioms. *Successor state axioms* are of the form $\Box(\bigcirc p(\boldsymbol{x}) \leftrightarrow (A_p^+ \vee (p(\boldsymbol{x}) \wedge \neg A_p^-)))$ with A_p^+ and A_p^- disjunctions of the form $\mathbf{do}(\alpha_1) \vee \ldots \vee \mathbf{do}(\alpha_m)$. Here, A_p^+ collects all actions that have $p(\boldsymbol{x})$ as effect and A_p^- all actions that have $\neg p(\boldsymbol{x})$ as effect. Intuitively, a successor state axiom expresses that $p(\boldsymbol{x})$ is the case in the next state iff an action is performed that has $p(\boldsymbol{x})$ as effect, or $p(\boldsymbol{x})$ is the case and no action with $\neg p(\boldsymbol{x})$ as effect is performed. To account for conditional effects a small modification is needed: If $(\neg)p(\boldsymbol{x})$ is an effect of α conditional on ϕ, then replace $\mathbf{do}(\alpha)$ by $\mathbf{do}(\alpha) \wedge \phi$ [5].

Now we are ready to define the components $\langle \Psi, \mathcal{L}_\Psi, \models_\Psi, \mathcal{M} \rangle$ of temporal GOAL. $\mathcal{L}_\Psi$ and $\models_\Psi$ are as defined above, given that $\mathcal{L} = \mathcal{L}_{LTL}$. $\mathcal{L}_\Psi$ thus allows temporal formulae inside the scope of the **B** and **G** operators, and we can use this to define several common sense notions of goals (see also [10]). Let **goal** χ be short for $\mathbf{G}\chi \wedge \neg\mathbf{B}\chi$. **goal** χ holds if χ is a goal and is not believed to occur inevitably, and corresponds more closely to the intuitive notion of a goal as being something that requires effort. When χ is of the form $\Diamond\chi'$ we say χ is an *achievement goal*. Note that we have $\mathbf{B}\chi \rightarrow \mathbf{G}\chi$ due to the rationality constraint $\Sigma \subseteq \Gamma$, which implies *realism* [6], but we do not have $\mathbf{B}\chi \rightarrow$ **goal** χ.

The set of mental states Ψ is restricted such that: knowledge bases only consist of PDDL axioms and precondition and frame axioms as defined above, belief bases are sets of literals, and goal bases are sets of temporal logic formulae. A rationality constraint is imposed on mental states $\langle K, \Sigma, \Gamma \rangle$ such that $(K \cup \Sigma) \subseteq \Gamma$ (cf. [10]). There are various reasons for this constraint, both conceptually as well as technically, but due to space restrictions we refer to [10] and only make two brief remarks here: The constraint implies that (i) K, Σ and Γ are mutually consistent, i.e., $K \cup \Sigma \cup \Gamma \not\models \bot$, which means that the agent cannot have something as a goal that is never realizable according to its beliefs, and (ii) statements believed to be currently the case are part of the goal base, which allows us to use the standard definition of progression of $\mathcal{L}_{LTL}$ formulae [2].

What remains is that we need to provide a definition of the mental state transformer $\mathcal{M}(\langle K, \Sigma, \Gamma \rangle, \alpha) = \langle K', \Sigma', \Gamma' \rangle$, i.e. we must specify how Σ' and Γ' can be obtained when α is performed (recall that knowledge bases do not change). We first specify how the belief base Σ' is obtained. To this end, the effects of performing α are collected in a set with *positive effects* $\mathit{Eff}^+_{LTL} = \{p(\bar{\mathbf{c}}) \mid K \cup \Sigma \cup \{\mathbf{do}(\alpha)\} \models \bigcirc p(\bar{\mathbf{c}})\}$ and a set with *negative effects* $\mathit{Eff}^-_{LTL} = \{\neg p(\bar{\mathbf{c}}) \mid K \cup \Sigma \cup \{\mathbf{do}(\alpha)\} \models \bigcirc \neg p(\bar{\mathbf{c}})\}$. Using these sets, and by slightly abusing notiation, we define:

$$\Sigma' = \Sigma \oplus (\mathit{Eff}^+_{LTL} \wedge \mathit{Eff}^-_{LTL}) \stackrel{df}{=} \mathit{Eff}^-_{LTL} \cup \mathit{Eff}^+_{LTL}$$

Finally, we need to specify Γ'. To do so, we use the progression operator from [2] *relative to the current belief base* Σ, but only specify some clauses of the inductive definition due to space limitations. For the base case $\phi \in \mathcal{L}_0$, $Progress(\phi, \Sigma) = \top$ if $\Sigma \models \phi$, otherwise $Progress(\phi, \Sigma) = \bot$. $Progress(\bigcirc\varphi, \Sigma) = \varphi$, the case that requires the constraint $\Sigma \subseteq \Gamma$ to be in place as it allows that a goal ϕ is entailed by the agent's beliefs. $Progress(\forall x.\varphi, \Sigma) = \bigwedge_{c \in \mathcal{C}} Progress(\varphi[c/x], \Sigma)$. We assume that $\bot \vee \chi$ is reduced to χ, $\bot \wedge \chi$ to $\bot$, $\top \vee \chi$ to $\top$, etc. In order to ensure progression always yields a consistent goal base Γ', some syntactic restrictions are imposed on the temporal formulae allowed in goal bases: they need to be in negation normal form (negation occurs only in front of atoms) and may not have occurrences of disjunction $\vee$ or the next operator $\bigcirc$.

Progression of a formula may result in $\bot$ which indicates that one of the goals has not been achieved, and, consequently, that the actions selected cannot be viewed as a plan for achieving these goals. To record such failure the special predicate *fail* has been introduced above, and is used to replace $\bot$ which also restores consistency. That is, the new goal base denoted by $Progress^{fail}(\Gamma, \Sigma)$ after performing an action is given by the set $\bigcup_{\varphi \in \Gamma} Progress(\varphi, \Sigma)$ where $\bot$, if present, has been replaced by *fail*. Finally, we define:

$$\mathcal{M}(\langle K, \Sigma, \Gamma \rangle, \alpha) = \langle K, \Sigma', Progress^{fail}(\Gamma, \Sigma) \rangle$$

where $\Sigma' = \mathit{Eff}^{-}_{LTL} \cup \mathit{Eff}^{+}_{LTL}$ and Eff^{+}_{LTL} and Eff^{-}_{LTL} are defined as above.

4 Compiling PDDL Problems into GOAL Agents

In this Section we show that a GOAL framework is expressive enough to define a planning problem from $PDDL_{Ax+CE+PC}$, given that such a framework is instantiated with a temporal logic KRT.

To compile a PDDL planning problem into a GOAL agent we use the concept of a compilation scheme. A *compilation scheme* is a mapping $\mathbf{F}$ from planning problems Π to GOAL agents $\mathbf{F}(\Pi)$ such that: (i) there exists a plan for Π iff there exists a run of $\mathbf{F}(\Pi)$ that achieves all goals, (ii) the translations of the initial and goal states of Π can be computed in polynomial time, and (iii) the size of $\mathbf{F}(\Pi)$ is polynomial in the size of Π. Compilation schemes in our sense are similar to that used in [13] but for obvious reasons differ in that we map to a GOAL agent instead of a planning problem in e.g. a different fragment of PDDL.

The compilation scheme $\mathbf{f}$ defined below maps every problem instance Π of the PDDL fragment $PDDL_{Ax+CE+PC}$ to a GOAL agent $\mathbf{f}(\Pi)$. The scheme $\mathbf{f}$ is defined by a tuple of functions $\langle f_{ax}, f_{at}, f_c, f_r, f_{as}, t_i, t_g, t_{pc} \rangle$ that map different parts of a PDDL problem $\langle \Delta, \mathcal{C}, \mathcal{I}, \mathcal{G}, \mathcal{PC} \rangle$ to corresponding GOAL agent $\langle K, \Sigma, \Gamma, R, A \rangle$ components.

Definition 14. *(Compiling PDDL Problems to* GOAL *Agents)*
Let $\Pi = \langle \Delta, \mathcal{C}, \mathcal{I}, \mathcal{G}, \mathcal{PC} \rangle$ *be a PDDL problem and* $\Delta = \langle \mathcal{L}_0, \mathcal{A}, \mathcal{O} \rangle$. *The scheme*

f *from PDDL problems to* GOAL *agents* $\mathbf{f}(\Pi) = \langle K, \Sigma, \Gamma, R, A\rangle$ *is defined by:*

- $K = f_{ax}(\mathcal{A}) \cup f_{at}(\mathcal{O}) \cup f_c(\mathcal{C})$,
- $\Sigma = t_i(\mathcal{I}) = \hat{\mathcal{I}}$,
- $\Gamma = t_g(\mathcal{G}) \cup t_{pc}(\mathcal{PC})$,
- $R = f_r(\Delta)$,
- $A = f_{as}(\Delta)$

The scheme **f** maps PDDL axioms into the knowledge base K by applying the well-known completion operator used to compute the completion of logic programms to these axioms, i.e. $f_{ax}(\mathcal{A}) = comp(\mathcal{A})$. Intuitively, the completion $comp(\mathcal{A})$ replaces implications with equivalences (cf. [1]). f_{at} maps the operators $\mathcal{O}$ to an action theory in $\mathcal{L}_{LTL}$ as discussed above that is also part of K. Finally, a *domain closure axiom* $f_c(\mathcal{C}) = \forall x.(x = c_1 \vee \ldots \vee x = c_n)$ is added to K, where $c_1, \ldots, c_n$ exhaust the constants in $\mathcal{C}$ (cf. [13]). The fact that a knowledge base of a GOAL agent is static corresponds with the fact that a PDDL domain does not change over time. The function t_i maps the initial state $\mathcal{I}$ to the belief base, such that $t_i(\mathcal{I}) = \hat{\mathcal{I}} = \{p(\bar{\mathbf{c}}) \in \mathcal{I}\} \cup \{\neg p(\bar{\mathbf{c}}) \mid \mathcal{I} \not\models p(\bar{\mathbf{c}}), p(\bar{\mathbf{c}}) \in \mathcal{B}\}$. t_g maps the goal $\mathcal{G} = \texttt{at end}(\phi_g)$ to $\Diamond\phi_g$. Assuming that $\mathcal{G} = \texttt{at end}\phi_g$ is the main goal, t_{pc} maps the planning constraints $\mathcal{PC}$ to the goal base as follows:

- $t_{pc}(\forall x.\varphi) = \forall x.t_{pc}(\varphi)$
- $t_{pc}(\texttt{always}\ \phi) = \phi$ **until** ϕ_g
- $t_{pc}(\texttt{sometime}\ \phi) = \phi$ **before** ϕ_g
- $t_{pc}(\texttt{at-most-once}\ \phi) = \phi$ **before** $\phi_g \rightarrow (\phi$ **until** $(\neg\phi$ **until** $\phi_g))$
- $t_{pc}(\texttt{sometime-after}\ \phi\ \phi') = \phi$ **before** $\phi_g \rightarrow (\phi$ **before** $(\phi'$ **before** $\phi_g))$
- $t_{pc}(\texttt{sometime-before}\ \phi\ \phi') = \phi$ **before** $\phi_g \rightarrow (\phi'$ **before** $(\phi$ **before** $\phi_g))$

This translation shows the difference between assuming a finite horizon as in planning that is not made in GOAL and has to be enforced by it. The function $f_r(\Delta)$ maps each of the actions $\alpha \in \mathcal{O}$ to an action rule **if** $\top$ **then** α in the program section, and f_{as} maps each operator definition $\langle \alpha, \pi_\alpha, \epsilon_\alpha\rangle \in \Delta$ to an action specification $\langle \alpha, \pi_\alpha, \top\rangle$. This works since effects are encoded in the action theories stored in the knowledge base.

Theorem 1. *There exists a solution for a PDDL problem* Π *iff there is a run* r *of the* GOAL *agent* $\mathbf{f}(\Pi)$ *such that for some* i *with* $r_i^m = \langle K, \Sigma, \Gamma\rangle$*:* $\Sigma \models \Gamma$.

Proof. Due to space limitations we only provide a sketch. The proof proceeds by first showing that the translation of the PDDL axioms and domain closure assumption is correct. Consecutively, we show that updates by performing actions in PDDL correspond with those derived from temporal action theories in GOAL. Finally, we show that there exists a plan (solution) iff the goal base of the compiled GOAL agent after performing the corresponding action sequence is empty and does not contain *fail*.

The theorem states that a planning problem can be represented by a GOAL agent. Another view on this result is that the meaning of the GOAL agent $F(\Pi)$ defines the *plan search space*. Obviously, part of this result is derived from the expressiveness of linear temporal logic, which allows to encode the semantics

of planning operators and plan constraints. However, GOAL agents do not use temporal logic to select actions but use action rules to do so. A *plan search space* thus is defined by the GOAL framework component of temporal GOAL (more specifically, Definition 11) and not by the temporal logic plugin.[1]

It is clear that each of the functions $\langle f_{at}, f_c, f_r, f_{as}, t_i, t_g, t_{pc} \rangle$ that define $\mathbf{f}$ execute in polynomial time, and, as a consequence, the size of the GOAL agent obtained by applying $\mathbf{f}$ is polynomial in the size of the original PDDL problem.

Corollary 1. $\mathbf{f}$ *defines a compilation scheme.*

It is clear that GOAL agents that are obtained by mapping a PDDL problem into GOAL can be translated back into a PDDL problem. This is not the case in general, however. For example, it is not clear how to map multiple achievement goals of the form $\Diamond\phi$ into a PDDL problem. Similarly, it is not clear how to map non-trivial action rules that use mental atoms of the form $\mathbf{G}(\varphi)$ into a PDDL problem. The question is which restrictions need to be imposed on GOAL agents to be able to map them into a PDDL problem. We propose the following restrictions as a *sufficient* set, but it remains to establish a *necessary* set:

1. The knowledge base consists of a *stratified* axiom set, and an LTL action theory.
2. The belief base is a set of literals $\hat{S}$ where S is a PDDL state.
3. The goal base consists of a *single* achievement goal of the form $\Diamond\phi$ with $\phi \in \mathcal{L}_0$ and, possibly, additional deadline goals of the form φ **until** ϕ and φ **before** ϕ.
4. The program section consists of *reactive* action rules only, i.e. rules that have only occurrences of belief atoms of the form $\mathbf{B}(\phi)$ with $\phi \in \mathcal{L}_0$ in their conditions.
5. Variables that occur in preconditions are parameters of the corresponding action.
6. All specified actions specified in the GOAL agent are *deterministic*. (PDDL does not allow for non-deterministic actions.)

Item (3) highlights one of the differences between dynamic agents, that may have multiple goals and dynamically adopt and/or drop goals, and static planning tasks. Alternatively, planners look for a fixed horizon determined by the goal state which defines a temporal window to which plan constraints apply, whereas this window needs to be made explicit in the mapping to GOAL.

5 Conclusion

We have shown that GOAL can be used as a planning formalism. To this end, the notion of a GOAL framework has been introduced that may be instantiated with various knowledge representation technologies. A GOAL framework defines the structure and semantics of a GOAL agent and has been introduced to be able to make a distinction between the expressiveness provided by GOAL itself and that provided by a KRT plugin. Temporal GOAL, a GOAL framework with a linear temporal logic plugin, has been used to compile planning problems with planning constraints into GOAL agents using so-called compilation schemes in the sense

[1] Another indication of this fact is that we do not need *compatibility axioms* as in [11].

of [13]. This paves the way for integrating a planner such as TLPlan [2] into GOAL and to combine the strenghts of planners with those of agent programming languages. We plan to integrate a planner into GOAL in combination with the notion of a module introduced in [9]. The idea is to introduce a variant called a planmodule where the context condition is used to define when to call a planner.

As argued, the semantics of GOAL may be viewed as defining a plan search space. One very interesting avenue for future research is how GOAL action rules with non-trivial mental state conditions can be used to reduce this search space. The idea is similar to how Golog programs [13] may reduce the search space and how heuristic knowledge can be used in TLPlan [2] and PDK [11].

References

1. K.R. Apt and R. Bol. Logic programming and negation: A survey. *Journal of Logic Programming*, 19:9–71, 1994.
2. F. Bacchus and F. Kabanza. Using temporal logics to express search control knowledge for planning. *Artificial Intelligence*, 16:123–191, 2000.
3. J. Baier and S. McIlraith. Planning with first-order temporally extended goals using heuristic search. In *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI06)*, pages 788–795, Boston, MA, July 2006.
4. R.H. Bordini, M. Dastani, J. Dix, and A. El Fallah Seghrouchni, editors. *Multi-Agent Programming: Languages, Platforms and Applications*. Springer, 2005.
5. S. Cerrito and M.C. Mayer. Using Linear Temporal Logic to Model and Solve Planning Problems. In F. Giunghiglia, editor, *Proc. of the Eigth Int. Conference on Artificial Intelligence (AIMSA'98)*, pages 141–152. Springer, 1998.
6. Philip R. Cohen and Hector J. Levesque. Intention Is Choice with Commitment. *Artificial Intelligence*, 42:213–261, 1990.
7. E.A. Emerson. Temporal and Modal Mogic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*. 1990.
8. A. Gerevini and D. Long. Plan constraints and preferences in PDDL3. Technical report, Department of Electronics for Automation, University of Brescia, 2005.
9. K.V. Hindriks. Modules as Policy-Based Intentions. In *Proceedings of the 5th International Workshop on Programming Multi-Agent Systems*, 2008.
10. K.V. Hindriks, M.B. van Riemsdijk, and W. van der Hoek. Agent programming with temporally extended goals. In *Proceedings of the Eighth International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2009)*, 2009.
11. M.C. Mayer, C. Limongelli, A. Orlandini, and V. Poggioni. Linear temporal logic as an executable semantics for planning languages. *Journal of Logic, Language and Information*, 16, 2007.
12. E.P.D. Pednault. ADL and the State-Transition Model of Action. *Journal of Logic and Computation*, 4(5):467–512, 1994.
13. G. Röger, M. Helmert, and B. Nebel. On the Relative Expressiveness of ADL and Golog. In *Proc. of the Eleventh Int. Conference on Principles of Knowledge Representation and Reasoning (KR 2008)*, pages 544–550. AAAI Press, 2008.
14. S. Sardina, L.P. de Silva, and L. Padgham. Hierarchical planning in BDI agent programming languages. In *Proc. of the Fifth Int. Conference of Autonomous Agents and Multi-Agent Systems (AAMAS 2006)*, pages 1001–1008, 2006.
15. Sylvie Thiébaux, Jörg Hoffmann, and Bernhard Nebel. In defense of PDDL axioms. *Artificial Intelligence*, 168:38–69, 2005.