Rich Goal Types in Agent Programming

Mehdi Dastani
Utrecht University
The Netherlands
mehdi@cs.uu.nl

M. Birna van Riemsdijk
Delft University of Technology
The Netherlands
m.b.vanriemsdijk@tudelft.nl

Michael Winikoff
University of Otago
New Zealand
michael.winikoff@otago.ac.nz

ABSTRACT

Goals are central to the design and implementation of intelligent software agents. Much of the literature on goals and reasoning about goals in agent programming frameworks only deals with a limited set of goal types, typically achievement goals, and sometimes maintenance goals. In this paper we extend a previously proposed unifying framework for goals with additional richer goal types that are explicitly represented as Linear Temporal Logic (LTL) formulae. We show that these goal types can be modelled as a combination of achieve and maintain goals. This is done by providing an operationalization of these new goal types, and showing that the operationalization generates computation traces that satisfy the temporal formula.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Intelligent agents, languages and structures*; I.2.5 [**Artificial Intelligence**]: Programming Languages and Software; F.3.3 [**Logics and Meaning of Programs**]: Studies of Program Constructs; D.3.3 [**Programming Languages**]: Language Constructs and Features

General Terms

Theory, Languages

Keywords

Agent Programming, Goals, Formal Semantics

1. INTRODUCTION

A widely-accepted approach to designing and programming agents is the *cognitive* approach, where agents are modeled in terms of mental concepts such as beliefs, goals, plans and intentions. Of the various concepts that have been used for cognitive agents, a key concept is *goals*. This is because agents are (by common definition) proactive, and goals are what allow agents to be proactive. It is also noteworthy that (the existence of explicitly represented) goals is one of the clearer differences between (proactive) agents and active objects. Goals have been extensively studied in artificial intelligence and multi-agent systems (e.g. [2, 13, 16, 21]).

Earlier work focused mostly on achievement goals, which represent a desired state that the agent wants to reach. However, increasingly other *goal types* are being studied such as maintenance goals, which represent a state the agent wants to maintain, and perform goals, which represent the goal to execute certain actions (e.g., [4, 7, 8, 11]). However, only considering a small number of goal types can be limiting, since in practical applications there may be goals that cannot be captured well by achievement or maintenance or perform goals.

To make this discussion more concrete, consider a personal assistant agent that manages a user's calendar and tasks. One goal the agent may have is booking a meeting. This would typically be modelled as an achievement goal that aims to bring about a state where all required participants have the meeting in their calendar. However, in practice, diaries change, and we want to ensure that the meeting remains in participants' diaries, and that should a key participant become unable to attend, a new time will be negotiated. This is not captured by an achievement goal. Rather, it is better modelled by a combined "achieve then maintain" goal which achieves a certain condition, and then maintains it over a certain time period. Another task that we might want the agent to undertake is to ensure that booking travel is not done until the budget is approved. Note that budget approval may be under the control (or perhaps just influence) of the agent, i.e. the agent may have plans for attempting to have the budget approved. Alternatively, it may be completely outside the agent's control, in which case the agent can just wait for it to happen and then enable the travel booking process.

A number of papers have taken this line of research a step further by taking arbitrary Linear Temporal Logic (LTL) formulae as goals [1, 2, 12, 13, 16], rather than considering specific goal types. The advantage of this approach is that it does not restrict the goal types that can be used. However, a possible disadvantage is that it requires extensive alterations of a more basic agent programming framework, the practical implications of which are not yet clear.

Temporal logic is also used by MetateM [10], but it is used directly for agent execution, whereas we use temporal logic as a design framework for specifying goal *types* which are mapped to existing implementations of achieve and maintenance goals. Additionally, MetateM requires a particular format for its rules: all rules are in one of the three forms: $\mathbf{start} \rightarrow \varphi$, or $\psi \rightarrow \bigcirc\varphi$ or $\psi \rightarrow \Diamond\phi$ where φ is a disjunction of literals, ψ is a conjunction of literals, and ϕ is a positive literal.

In this paper, we propose an approach that is somewhere in between those focusing on a limited set of goal types and those in which arbitrary LTL formulae can be taken as goals. We propose an approach in which goals that are represented by relatively complex LTL formulae are operationalized by *translating* these LTL formulae to more basic achieve and maintain goals. The advantage of this approach is that the goal types can be integrated in existing

Cite as: Rich Goal Types in Agent Programming, M. Dastani, M.B. van Riemsdijk, M. Winikoff, *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, Yolum, Tumer, Stone and Sonenberg (eds.), May, 2–6, 2011, Taipei, Taiwan, pp. XXX–XXX.

agent programming frameworks that already have an operationalization of achieve and maintain goals. We illustrate the approach by showing how a number of LTL formulae can be translated to achieve and maintain goals.

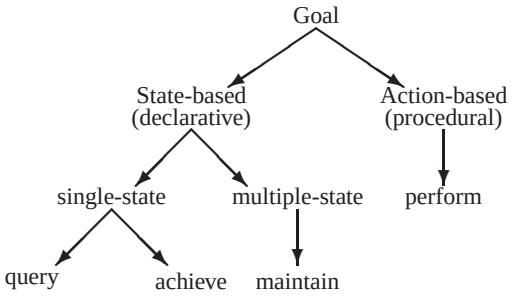
This paper builds on previous work [19] which presented a unifying framework for goals based on viewing goals as LTL formulae that described desired progressions. This previous work captured existing basic goal types, whereas we propose a framework that allows the use of richer goal types. Section 2 provides a brief description of the unifying framework on which we build. Section 3 presents the new goal types which are formalized and realised in sections 4 and 5. Finally, in Section 6 we conclude the paper and discuss some future directions.

2. A UNIFYING FRAMEWORK FOR GOAL TYPES

This paper builds on the framework of van Riemsdijk *et al.* [19] in which a goal type is informally considered as a property characterising a set of computation traces. The framework is explained in terms of an abstract architecture for operationalizing goals such as achieve and maintain goals. In particular, the operationalized architecture aims to capture essential aspects of goals in agent programming frameworks in terms of properties of computation traces, abstracting from particular goal types.

The framework models goals as follows. The state includes state-related propositions, which capture the state of the world, as well as propositions of the form $done_i(a)$ which capture the performance of actions. This approach takes an abstract view of a goal type as a particular pattern, or structure, of a formula in Linear Temporal Logic (LTL). A given LTL formula corresponds to a set of traces which make it true, and is viewed as a goal in that it allows a given system trace to be classified as satisfying the goal or not.

van Riemsdijk *et al.* [19] defined four commonly-used goal types: achievement goal, (reactive) maintenance goal[1], perform goal, and query goal. These goals are classified into a taxonomy (below), and an operationalization is provided for them within a single framework, using a simple execution cycle which was extended with additional rules of the form $\langle condition, action \rangle$ where actions could be to SUSPEND, ACTIVATE or DROP a goal.



A key feature of this approach is that it balances flexibility with being structured enough to ensure desired properties of goals, and hence for it to make sense for the resulting constructs to be called "goals". A range of desired properties of goals have been identified in the literature. Winikoff *et al.* [21, Section 2] survey existing literature and, based on this, identify the following desired properties of goals:

1. Persistent: goals should only be dropped for a good reason.
2. Unachieved: goals should not already hold; alternatively, they are dropped when they are achieved.
3. Possible: goals should be dropped when they are impossible to achieve.
4. Known: the agent should be aware of its goals. In implementation terms, this implies a measure of reflectiveness.
5. Consistent: goals should not be in conflict with other adopted goals.

A number of additional properties are identified by [4]. Some of these (Producible/Terminable and Suspendable) simply correspond to the existence of a goal life-cycle. The others (Variable Duration and Action Decoupled) relate to the notion of long term goals that they argue for.

One advantage of the proposed goals framework is that the properties representing a specific goal type can be dealt with in a general and systematic way. Suppose we have a goal ϕ of a certain type. The framework maps ϕ to a goal construct $\mathsf{g}(R, \ldots)$ where R is a collection of rules (derived from ϕ) that govern transitions between goal states. We can then show that the goal's realization meets the properties of being persistent, unachieved, and possible, by requiring that the operationalization of $\mathsf{g}(R, \ldots)$ results in the goal being dropped exactly when ϕ becomes known to be true, or known to be impossible. For example, consider the goal to achieve p. This is mapped [19, Section 3.3.1] to $\mathsf{g}(R, \ldots)$ where R consists of two rules: one to activate the goal when it is adopted, and one to drop the goal when $s \vee f$ becomes true, where s is the "success condition", i.e. when the goal is succeeded, here $s = p$; and f is a description of a condition under which the goal becomes impossible to achieve. Here f depends on properties of p, but may be simply false, if it always remains possible to achieve p. Then it is straightforward to show that, under the operational semantics for goals defined by [19], the goal $\mathsf{g}(R, \ldots)$ is dropped exactly when p becomes known to be true, or known to be impossible (properties 1–3, above). The property of being known (property 4) is achieved by having an explicit goal base which allows the agent to reflect on which goals it has. Consistency (property 5) concerns interactions between goals, and is beyond the scope of this paper.

3. NEW GOAL TYPES

In this paper, goals are represented explicitly as specific formulae in Linear Temporal Logic (LTL) [9]. While [19] defined the notion of goal as representing preferred progressions, and informally referred to LTL to explain this, the operationalization itself did not use LTL explicitly in the representation of goals. The LTL formulae that we use to represent goals are defined by the following grammar. In addition to basic propositions (p), and standard propositional connectives, it has the temporal connectives $\Diamond$ ("eventually"), $\Box$ ("always"), and U ("until"). We use standard abbreviations such as $\top \equiv p \vee \neg p$ and $\phi_1 \vee \phi_2 \equiv \neg((\neg\phi_1) \wedge (\neg\phi_2))$. Note that we do not use the next ($\bigcirc$) connective because the goals we consider in this paper does not use this operator.

$$\phi \ ::= p \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \Diamond\phi \mid \Box\phi \mid \phi_1 \mathsf{U} \phi_2$$

The semantics are the usual ones (given below). They are defined over a model $\mathcal{M}$ which is an infinite sequence of states, where each state is a set of propositions that hold in that state. A formula ϕ is true with respect to a model $\mathcal{M}$ and an index i, indicating the

[1]Maintenance goals are defined as being *proactive* where violation of desired state is anticipated and avoided, or *reactive*, where the desired state is "recovered" once it is violated [8].

current state. We use $\mathcal{M}_i$ to denote the ith state in $\mathcal{M}$.

$$\begin{aligned}
\mathcal{M}, i \models p \quad & \text{iff} \quad p \in \mathcal{M}_i \\
\mathcal{M}, i \models \neg\phi \quad & \text{iff} \quad \mathcal{M}, i \not\models \phi \\
\mathcal{M}, i \models \phi_1 \wedge \phi_2 \quad & \text{iff} \quad \mathcal{M}, i \models \phi_1 \text{ and } \mathcal{M}, i \models \phi_2 \\
\mathcal{M}, i \models \Diamond\phi \quad & \text{iff} \quad \exists k \geq i : \mathcal{M}, k \models \phi \\
\mathcal{M}, i \models \Box\phi \quad & \text{iff} \quad \forall k \geq i : \mathcal{M}, k \models \phi \\
\mathcal{M}, i \models \phi_1 \mathcal{U} \phi_2 \quad & \text{iff} \quad \exists k \geq i : \mathcal{M}, k \models \phi_2 \text{ and} \\
& \quad \forall j \text{ such that } i \leq j < k : \mathcal{M}, j \models \phi_1
\end{aligned}$$

We now extend the taxonomy of [19] with additional goal types, including those discussed in the introduction. Specifically, the personal assistant agent example introduced new goal types. The first, booking a meeting and then maintaining participant availability, can be formalised as follows. Let *pa* be short for *participantsAvailable*, *ms* be short for *meetingScheduled*, and *mo* be short for *meetingOccurs*, then we represent the goal of booking a meeting as the following LTL formula:

$$\Diamond(ms \wedge (pa \mathcal{U} mo))$$

Considering the second goal, not booking travel until the budget has been approved, this can be formalised as[2]:

$$(\neg bookTravel) \mathcal{U} budgetApproved$$

Abstracting from the specific goal instances to general goal types, we have defined goal types of the form $\Diamond(\phi_1 \wedge (\phi_2 \mathcal{U} \tau))$ (achieve ϕ_1 and then maintain ϕ_2 until τ), and $\phi \mathcal{U} \tau$ (maintain ϕ until τ). We now generalise these goal types by considering a range of ways in which a (non-temporal) property ϕ can be required to hold over a number of states.

Consider a multiple-state goal where a (non-temporal) property ϕ is required to hold over a number of states in the trace. The taxonomy in the previous section (from [19]) only supports a single multiple-state goal. However, there are a number of ways in which a goal pattern can apply to a sequence of states. It can apply:

1. to all states: $\Box\phi$;
2. at the start of the trace: $\phi \mathcal{U} \tau$, where τ is a formula that describes the state at which ϕ is no longer required to be true;
3. at the end of the trace: $\Diamond(\tau \wedge \Box\phi)$, where τ is a "trigger" formula that describes the state at which ϕ begins to be required to be true; or
4. in the middle of the trace: $\Diamond(\tau \wedge (\phi \mathcal{U} \tau'))$, where τ is the starting trigger and τ' the ending trigger; or
5. it can apply to a number of sub-sequences of states: $\Box(\tau \rightarrow (\phi \mathcal{U} \tau'))$, where τ is a trigger that describes a state at which ϕ begins to be required to hold, and τ' describes the states at which ϕ is no longer required to hold.

These cases for multiple-state goal types are summarised in Figure 1. Note that in all cases we require that ϕ hold at all states within the specified region.

Considering the personal assistant example, the first goal $\Diamond(ms \wedge (pa \mathcal{U} mo))$ corresponds to the fourth case above and the second goal corresponds to the second case above. Note that we could also consider a goal where $\neg bookTravel$ must hold on different sequences of states, e.g. that once there is no more money in the budget (*nmm*) then travel cannot be booked until a (new) budget is approved, formally: $\Box(nmm \rightarrow (\neg bookTravel \mathcal{U} budgetApproved))$.

[2]This goal would be expected to be used in conjunction with a goal to book travel, $\Diamond bookTravel$.

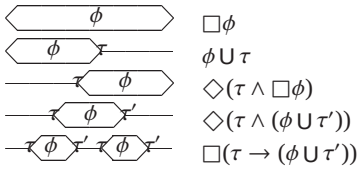


Figure 1: Multiple-state goal types

We thus define the possible LTL patterns that we allow as multiple-state goal types as follows, where ϕ and τ are propositional (i.e. non-temporal). Instead of only supporting a single type of multiple-state goal, as in previous work, we support the following, which correspond with the cases in Figure 1:

$$G_m ::= \Box\phi \mid \phi \mathcal{U} \tau \mid \Diamond(\tau \wedge \Box\phi) \mid \Diamond(\tau \wedge (\phi \mathcal{U} \tau')) \mid \Box(\tau \rightarrow (\phi \mathcal{U} \tau'))$$

We also allow the single-state goal type $\Diamond\phi$. We would like to emphasize that the proposed temporal goal types are by no means exhaustive and that other LTL patterns can be identified to represent other multiple-state goal types as well. Our claim is that the proposed goal types are intuitive in that the corresponding LTL patterns represents desirable execution traces as illustrated by the examples, and that they can be operationalized by means of achieve and maintain goals. We also would like to note that other goal types (e.g., those mentioned in Figure 1) can be represented in our framework as well. For example, the query goal can be represented as $(Bp) \vee (B\neg p)$ (where Bp denotes that agent believes p in the current state), the achieve goal as $\Diamond p$, the maintenance goal as $\Box p$, and the perform goal as $done(a)$, where $done(a)$ denote the fact that action a is performed.

4. REALISING THE NEW GOAL TYPES

The most characterising feature of our programming approach is to represent goals explicitly as temporal formulae and to operationalize these formulae in terms of achieve and maintain goals. The advantage of this approach is that achieve and maintain goal types have already well-defined operational semantics in some of the existing agent programming frameworks (e.g., 2APL [6], Jadex [15], and JACK [5]) such that our goal types can be used to extend these frameworks.

In order to realise the new goal types in an operational setting, we assume that an agent configuration comprises a belief base, consisting of propositional atoms, and two goal bases. The first goal base, called the *temporal goal base*, contains goals specified by temporal LTL formulae. The second goal base, called the *basic goal base*, consists of achieve goals of the form $A(\phi)$ (read as: ϕ should be achieved) and maintain goals of the form $M(\phi, \tau)$ (read as: ϕ should be maintained until τ), where ϕ and τ are propositional formulae. The maintain goal $M(\phi, \bot)$ represents that ϕ should be maintained indefinitely.

The operationalization of temporal goal types can then be defined in terms of operations on temporal and basic goal bases. In this paper, we assume that the achieve and maintain goals have a correct operationalization. In particular, for the achieve goal we assume that if $A(\phi)$ is in the basic goal base, then the agent belief base will eventually entail ϕ, and for the maintain goals we assume that if $M(\phi, \tau)$ is in the basic goal base, then the agent belief base entails ϕ until τ is entailed by the belief base. The latter assumption thus ensures that there is no need for reachieving ϕ (typically referred to as reactive maintain goal), since we assume it does not become false once the basic maintain goal has been adopted. The

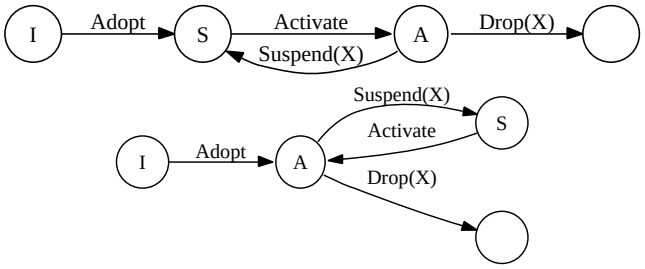


Figure 2: Temporal (top) and basic (bottom) goal life cycles

basic maintain goal is thus interpreted as proactive avoiding the violation of desired states. Clearly, our assumption for the achieve goal is realistic as many agent programming languages such as 2APL [6] provide already programming constructs to implement achieve goals with the correct interpretation. Also, our assumption for the maintain goal is realistic as there exist operationalization proposals for the maintain goals [11] that perform lookahead steps to avoid generating execution paths where the goal is violated.

For the purposes of this paper we want to show that our framework realises temporal goals correctly if we have correct operationalization of achieve and maintain goals. The idea is that the temporal goal $\Diamond\phi$ can be operationalised by adding the achieve goal $A(\phi)$ to the basic goal base. Similarly, the temporal goal $\phi \mathcal{U} \tau$ can be operationalised by adding the maintain goal $M(\phi, \tau)$ to the basic goal base. More complex temporal goals can be operationalized by adding both achieve and maintain goals to the basic goal base, as will be shown in the sequel.

In our framework, goals have a life cycle as illustrated in Figure 2. Both temporal (top) and basic (bottom) goals begin in an initial state (I), and can then be either in active (A) or suspended (S) states. For the reasons explained in the next section, a temporal goal can be adopted entering in a suspended state and a basic goal (either an achieve or a maintain goal) can be adopted entering in the active state. A temporal goal in a suspended state can be activated after which it can be either suspended again or dropped. A basic goal in an active state can be either suspended or dropped.

While temporal goals are assumed to be given by an agent program (specified by an agent programmer), the basic (achieve and maintain) goals are adopted as a consequence of processing temporal goals. In our framework, a temporal goal is dropped if it can be satisfied by adopting an achieve or a maintain goal. A temporal goal is suspended if it can be *partially* satisfied by adopting an achieve or a maintain goal. For this reason, the actions $Drop(X)$ and $Suspend(X)$ drop and suspend the corresponding temporal goals and, at the same time, add the basic goal X to the basic goal base. This notation should not be confused and read as parameterised drop and suspend actions. Finally, the actions $Drop(\emptyset)$ and $Suspend(\emptyset)$ remove a temporal goal from temporal goal base and leave the basic goal base unchanged.

4.1 Life Cycle for Temporal Goals

In order to specify the state transitions of temporal goals, a set of condition-action pairs is assigned to each temporal goal. The condition of such a pair is a test on an agent's belief base and the action is to change the goal's state. The condition-action pairs can be either generic or domain related. A generic condition-action pair specifies a transition for all instances of a goal type while a domain related condition-action pair specifies a transition for a specific instance of a goal type depending on the application at hand.

ϕ	$\delta_g(\phi)$
$\Diamond\phi$	$\{\langle\phi, Drop(\emptyset)\rangle\ ,\ \langle\neg\phi, Drop(A(\phi))\rangle\}$
$\Box\phi$	$\{\langle\phi, Drop(M(\phi,\bot))\rangle\ ,\ \langle\neg\phi, Suspend(A(\phi))\rangle,$ $\langle\phi, Activate\rangle\}$
$\phi \mathcal{U} \tau$	$\{\langle\phi, Drop(M(\phi,\tau))\rangle\ ,\ \langle\neg\phi, Suspend(A(\phi))\rangle,$ $\langle\phi, Activate\rangle\}$
$\Diamond(\tau \wedge \Box\phi)$	$\{\langle\phi\wedge\tau, Drop(M(\phi,\bot))\rangle\ ,$ $\langle\neg(\phi\wedge\tau), Suspend(A(\phi\wedge\tau))\rangle,$ $\langle\phi\wedge\tau, Activate\rangle\}$
$\Diamond(\tau \wedge (\phi \mathcal{U} \tau'))$	$\{\langle\phi\wedge\tau, Drop(M(\phi,\tau'))\rangle\ ,$ $\langle\neg(\phi\wedge\tau), Suspend(A(\phi\wedge\tau))\rangle,$ $\langle\phi\wedge\tau, Activate\rangle\}$
$\Box(\tau \rightarrow (\phi \mathcal{U} \tau'))$	$\{\langle\tau\wedge\phi, Suspend(M(\phi,\tau'))\rangle\ ,$ $\langle\tau\wedge\phi, Activate\rangle\}$

Figure 3: Generic Condition-Action Pairs for Temporal Goals

The domain related condition-action pairs can be used for various purposes, e.g., to activate temporal goals in domain specific situations in which the goals are likely to be realised (in addition to the generic ones) or to suspend them in situations in which the goals are not likely to be realised (in addition to the generic ones). These domain related condition-action pairs should be designed carefully since otherwise they may cause undesirable behavior. We assume domain related condition-action pairs are assigned to each temporal goal by the agent programmer in order to influence the life cycle of temporal goal based on domain dependent knowledge. We use $\delta_g(\phi)$ to refer to the set of generic condition-action pairs of temporal goal ϕ as specified in Figure 3.

It is important to note that these condition-action pairs are only applicable when goals are in specific states, e.g., goals can only be dropped when they are active and activated when they are suspended. The suspension condition-action pairs can not only fire in the active state, but also when a goal is adopted (which moves the goal into the suspended state). The applicability of condition-action pairs are formally specified by the transition rules in Section 5.1.

The first temporal goal type is characterised as $\Diamond\phi$, where ϕ is a non-temporal formula. The first generic condition-action pair assigned to this goal indicates that when ϕ is entailed by the agent's beliefs in its current state, then the goal $\Diamond\phi$ can be dropped and no basic goal is added to the basic goal base. In this case, the temporal goal is already believed to be satisfied. The second condition-action pair indicates that if ϕ is not entailed by the agent's beliefs in its current state, then the goal $\Diamond\phi$ can be dropped, but simultaneously the agent adopts the achieve goal $A(\phi)$ by adding it to its basic goal base. Our assumption that the achieve goal is operationalized correctly ensures that the temporal goal $\Diamond\phi$ will be satisfied by the agent execution. As we will see later on, these drop actions only take place if the temporal goal is in its active state. It should be noticed that there is no condition-action pair to activate this temporal goal. We assume such a condition-action pair is added as a domain related pair by the programmer. An example of such a pair is $\langle\top, Activate\rangle$, which is a strong activation condition as it indicates that the temporal goal should always be activated.

The second temporal goal is characterised by $\Box\phi$, where ϕ is a non-temporal formula. The first condition-action pair drops the temporal goal and adopts the maintain goal, adding it at the same time to the basic goal base. Again, our assumption of correct operationalisation of maintain goals ensures that the temporal goal will be satisfied, which is why we can drop the temporal when adopting the basic maintain goal. That is, there is no need to suspend and

reactivate this temporal goal should ϕ no longer be believed, since the latter is assumed not to occur once the basic maintain goal is adopted. Note that a maintain goal $M(\phi, \bot)$ is adopted in a state that satisfies the maintain condition ϕ. The second condition-action pair indicates that the temporal goal $\Box\phi$ should be suspended if ϕ is not entailed by the current agent's beliefs, while adopting the achieve goal $A(\phi)$ to pursue a state from which the maintain goal can be maintained. This can occur only when adopting this temporal goal while ϕ does not hold. Since this temporal goal is dropped after activation, it cannot be suspended again from the active state. The third pair ensures that the temporal goal is activated again upon achievement of ϕ.

The operationalisation of the third temporal goal type is very much similar to the second one since $\Box\phi$ is equivalent to $\phi \mathsf{U} \bot$, i.e., the only difference is that ϕ should be maintained until τ holds, and thus not indefinitely as was the case with $\Box\phi$.

The fourth temporal goal type is characterised by $\Diamond(\tau \wedge \Box\phi)$. The first condition-action pair ensures that ϕ is maintained indefinitely if the agent's current beliefs entails $\phi \wedge \tau$. However, if either ϕ or τ do not hold, then the temporal goal is suspended and the achieve goal $A(\phi \wedge \tau)$ is added to the basic goal base. This ensures that the agent will pursue the condition before maintaining ϕ indefinitely. The third pair ensures that the goal is activated again once the condition $\phi \wedge \tau$ is satisfied. We adopt a goal to achieve both ϕ and τ, and not just τ, because in the state in which the agent transitions from achieving τ to maintaining ϕ, we want ϕ to be true (so it can be maintained). This is also the reason why the third condition-action pair has a condition of $\phi \wedge \tau$, and not just τ. These points also apply to the following temporal goal types.

The fifth temporal goal type is characterised by $\Diamond(\tau \wedge (\phi \mathsf{U} \tau'))$. The first condition-action pair indicates that if $\phi \wedge \tau$ holds, then the temporal goal can be dropped and at the same time the basic maintain goal $M(\phi, \tau')$ is adopted. Note that if $\phi \wedge \tau$ holds, the maintain goal $M(\phi, \tau')$ ensures that ϕ is maintained until τ' is achieved. The second condition-action pair covers the situation where either ϕ or τ does not hold. In such a situation, the temporal goal cannot be realized. The temporal goal is therefore suspended, but the basic achievement goal $A(\phi \wedge \tau)$ is added in order to achieve the condition for the first condition-action pair and the goal is activated again once this condition is satisfied.

Finally, the sixth temporal goal type is characterized by $\Box(\tau \rightarrow (\phi \mathsf{U} \tau'))$. The first condition-action pair specifies that when $\phi \wedge \tau$ is entailed by an agent's belief base the temporal goal can be suspended (not dropped) and the basic maintain goal $M(\phi, \tau')$ adopted. The adopted maintain goal ensures the maintenance of ϕ until τ'. Note that the temporal goal can be pursued again since it was suspended, instead of being dropped. The second condition-action pair is to activate the temporal goal. Note that the condition of this pair is the same as the condition of the first pair, but that the first pair is applicable only to active goals and the second only to suspended goals. However, in order to avoid a loop (suspend, activate, suspend, ...) we impose an additional condition that a suspended goal of this type cannot be activated again until the plan generated for the goal has been performed. Formalising this restriction is straightforward but is omitted for space reasons.

4.2 Life Cycle for Basic Goals

Generic condition-action pairs are also assigned to basic goals when they are adopted for a temporal goal. As we will see later, these condition-action pairs implement the generic relation between beliefs and goals, e.g., an achieve goal $A(\phi)$ is dropped when ϕ is believed and a maintain goal $M(\phi, \tau)$ is dropped if τ is believed.

Note that the second condition-action pair for basic maintenance goals, which activates the goal, is only applicable to suspended goals (as defined in the transition rules in Section 5.2). On the other hand, the last condition-action pair, which drops a basic maintenance goal, can only be applied to an active basic maintenance goal. It should also be observed that our assumption about correct operationalisation of maintain goals implies that the condition of the suspend action is never satisfied, and therefore the basic maintain goal never gets into the suspended state. This condition-action pair is used in cases where the assumption that maintenance goals are correctly operationalised does not hold.

BGoal	**Condition – Action Pairs** λ
$A(\phi)$	$\{\langle \phi, Drop \rangle\}$
$M(\phi, \tau)$	$\{\langle \neg\phi \wedge \neg\tau, Suspend \rangle, \langle \phi \vee \tau, Activate \rangle, \langle \tau, Drop \rangle\}$

Figure 4: Generic Condition-Action Pairs for Basic Goals

Additionally, a set of domain related condition-action pairs are assigned to each basic goal. Like temporal goals, domain related condition-action pairs for basic goals should be used with care since otherwise they may cause undesirable behavior. The condition-action pairs for basic goals may seem redundant as they do not contribute to the operationalisation of temporal goals. However, these condition-action pairs generalise the presented framework, thus allowing for the design of arbitrary basic goal behaviors, which may be useful for a variety of domain dependent applications. For example, a robot with an achieve goal to be at a certain position will suspend its achieve goal when its battery charge is not sufficient.

5. OPERATIONAL SEMANTICS

The operationalization of goal types is accomplished by operational semantics, which indicate possible transitions between agent configurations due to goal processing.

Definition 1 *The agent configuration is defined as a tuple* $\langle \sigma, \gamma_t, \gamma_b \rangle$, *where* σ *is the agent's belief base (a finite set of propositional atoms),* γ_t *is the temporal goal base (a finite set of triples of the form* $(\phi, state, \Delta)$ *where* ϕ *is a temporal formula, state is the state of the temporal goal (init, active or susp(ended)), and* Δ *is the set of condition-action pairs governing the life cycle), and* γ_b *is the basic goal base (a finite set of triples of the form* $(g, state, \Delta)$ *where g is one of* $A(\phi)$ *or* $M(\phi, \tau)$, *or* $M(\phi, \bot)$, *and the remaining components are the same as for the temporal goal base).* $A(\phi)$ *and* $M(\phi, \tau)$ *denote goals to achieve and maintain the non-temporal formula* ϕ, *respectively. The maintain goal* $M(\phi, \tau)$ *has an additional argument, a proposition* τ, *that indicates the deadline until which* ϕ *should be maintained.*

The operational semantics defines how the agent pursues complex temporal goals in terms of the pursuit of basic achievement and maintenance goals. How the agent pursues basic achievement and maintenance goals is not defined here, and can be found elsewhere (e.g. [19]). We make assumptions about the pursuit of achievement and maintenance goals being operationalised correctly, and then show that, given these assumptions, the semantics given here correctly achieve complex temporal goals.

5.1 Transition Rules for Temporal Goals

Below, we specify transition rules for individual temporal goals, and after that transition rules that lift these to sets of temporal goals. Let ϕ be a temporal goal, δ_d a set of domain related condition-action pair for ϕ, and $\delta_g(\phi)$ be the set of generic condition-action pairs as defined in Section 4.1. Let λ be the generic condition-action pair

for basic goals as defined in Figure 4. We define $+(X, \gamma_b)$ (i.e., γ_b extended with basic goal X) as follows (note that we do not add any domain related condition-action pair to basic goals).

- $+(\emptyset, \gamma_b) = \gamma_b$
- $+(X, \gamma_b) = \gamma_b \cup \{(X, active, \lambda)\}$

The first two rules below (*Adopt*1 and *Adopt*2) define the adoption of a temporal goal $(\phi, init, \delta_d)$, firstly in the case where there is a condition-action pair to suspend the goal while adopting a basic goal, and secondly where there is no applicable condition-action pair to suspend the goal (no basic goal is added). Note that in both cases, temporal goals are adopted entering in the suspended state. Temporal goals are initially suspended in order to ensure that their corresponding maintain goal is added to the basic goal base only in states where their maintain condition is satisfied. This can be verified by observing the condition-action pairs that add maintain goals to the basic goal base in Figure 3. Note that the application of the first transition rule adds a maintain goal to the basic goal base only for the sixth goal type and only when the condition of the to be added maintain goal is satisfied. In other cases, the temporal goals that would add a maintain goal are suspended without adding the maintain goal to the basic goal base until the maintain conditions are satisfied.

In contrast to temporal goals, basic goals are adopted in an active state (see above in the definition of $+(X, \gamma_b)$). This is because achieve goals can always be activated, i.e., there is no reason why they should be suspended at the start. A maintain goal can also start in an active state since its adoption condition ensures its maintain condition holds as explained above.

It is also important to notice that in the first two transition rules we use generic condition-action pairs only from δ_g (and not from Δ). This is because domain related condition-action pairs are only used for activation and dropping of the goals, not for their adoption. Finally, observe that the first five transition rules allow transitions from one single temporal goal to a set of temporal goals. This means that these transition rules cannot be applied consecutively. However, the last transition rule is designed to manage the processing of a set of temporal goals in terms of transitions that are derivable from the first five transition rules.

$$\frac{\langle c, Suspend(X)\rangle \in \delta_g(\phi) \quad \sigma \models c}{\langle \sigma, (\phi, init, \delta_d), \gamma_b\rangle \rightarrow \langle \sigma, \{(\phi, susp, \delta_d \cup \delta_g(\phi))\}, +(X, \gamma_b)\rangle} \; Adopt1$$

$$\frac{\neg\exists\langle c, Suspend(X)\rangle \in \delta_g(\phi) : \sigma \models c}{\langle \sigma, (\phi, init, \delta_d), \gamma_b\rangle \rightarrow \langle \sigma, \{(\phi, susp, \delta_d \cup \delta_g(\phi))\}, \gamma_b\rangle} \; Adopt2$$

The following rule activates a suspended temporal goal.

$$\frac{\langle c, Activate\rangle \in \Delta \quad \sigma \models c}{\langle \sigma, (\phi, susp, \Delta), \gamma_b\rangle \rightarrow \langle \sigma, \{(\phi, active, \Delta)\}, \gamma_b\rangle} \; Activate$$

The following rule suspends an active temporal goal and (possibly) adds a basic goal to the basic goal base.

$$\frac{\langle c, Suspend(X)\rangle \in \Delta \quad \sigma \models c}{\langle \sigma, (\phi, active, \Delta), \gamma_b\rangle \rightarrow \langle \sigma, \{(\phi, susp, \Delta)\}, +(X, \gamma_b)\rangle} \; Suspend$$

The following rule drops a temporal goal and (possibly) adds a basic goal to the basic goal base.

$$\frac{\langle c, Drop(X)\rangle \in \Delta \quad \sigma \models c}{\langle \sigma, (\phi, active, \Delta), \gamma_b\rangle \rightarrow \langle \sigma, \{\}, +(X, \gamma_b)\rangle} \; Drop$$

The following transition rule specifies how the above rules for single temporal goals (denoted as g) can be lifted to temporal goal bases. Note that whereas g is a single goal (a tuple), g' is a set of goals (either singleton or empty).

$$\frac{g \in \gamma_t \quad \langle \sigma, g, \gamma_b\rangle \rightarrow \langle \sigma, g', \gamma_b'\rangle}{\langle \sigma, \gamma_t, \gamma_b\rangle \rightarrow \langle \sigma, (\gamma_t \setminus \{g\}) \cup g', \gamma_b'\rangle} \; Lift$$

5.2 Transition Rules for Basic Goals

The following rules specify the life cycle of a basic goal X. The *DropBasic* rule indicates that if a basic goal is in an active state, then it can be dropped if there is a corresponding condition-action pair for which the condition is satisfied in the current state and the action is a drop action. It should be noted that for a basic achievement goal we have $\langle \phi, Drop\rangle$, which indicates that the basic goal $A(\phi)$ can be dropped when ϕ holds. Similarly, for a basic maintain goal we have $\langle \tau, Drop\rangle$, which indicates that the maintain goal $M(\phi, \tau)$ can be dropped if τ holds in the current state.

$$\frac{\langle c, Drop\rangle \in \Delta \quad \sigma \models c}{\langle \sigma, \gamma_t, (X, active, \Delta)\rangle \rightarrow \langle \sigma, \gamma_t, \{\}\rangle} \; DropBasic$$

The following transition rule ($SuspB$) specifies that a goal in an active state can be suspended. Note that the corresponding condition-action pair for a maintain goal indicates that a maintain goal can be suspended if neither ϕ nor τ hold in the current state.

$$\frac{\langle c, Suspend\rangle \in \Delta \quad \sigma \models c}{\langle \sigma, \gamma_t, (X, active, \Delta)\rangle \rightarrow \langle \sigma, \gamma_t, \{(X, susp, \Delta)\}\rangle} \; SuspB$$

The next transition rule is designed to manage the transition of a basic goal from suspended to active state. Note that the corresponding condition-action pair for a maintain goal states that the basic maintain goal $M(\phi, \tau)$ can be activated if either ϕ or τ hold in the current state.

$$\frac{\langle c, Activate\rangle \in \Delta \quad \sigma \models c}{\langle \sigma, \gamma_t, (X, susp, \Delta)\rangle \rightarrow \langle \sigma, \gamma_t, \{(X, active, \Delta)\}\rangle} \; ActivB$$

Finally, as for temporal goals, we have a transition rule that specifies how the above rules for single basic goals (denoted as g) can be lifted to basic goal bases. Note again that whereas g is a single basic goal (a tuple), g' is a set of basic goals (either singleton or empty).

$$\frac{g \in \gamma_b \quad \langle \sigma, \gamma_t, g\rangle \rightarrow \langle \sigma, \gamma_t, g'\rangle}{\langle \sigma, \gamma_t, \gamma_b\rangle \rightarrow \langle \sigma, \gamma_t, (\gamma_b \setminus \{g\}) \cup g'\rangle} \; LiftB$$

5.3 Properties

In the following, we use P to denote the set of non-temporal propositional formulae, and we use $\models$, $\models_{cwa}$, and $\models_{LTL}$ to respectively denote propositional entailment, propositional entailment based on the closed-world assumption, and LTL entailment.

We define the transition system Σ to include the rules defined earlier in this section. It is important to observe that the transition system Σ contains other transition rules in addition to those presented in sections 5.1 and 5.2. In particular, Σ is assumed to contain transition rules for action execution, which may change the belief states. As the application of transition rules may be interleaved, possible belief changes may influence the goal life cycles. One assumption that we make about the details of transition system Σ, is that the rules defined earlier in this section have a higher priority than other rules. So for instance, if there are two applicable transition rules, say one for deriving/allowing an action execution transition, and the other for the application of a condition-action pair of a goal, then the second transition rule is applied to derive/allow the transition of goal life cycle. This assumption is crucial to show the properties of our transition system in the rest of this section.

Definition 2 *Let $\mathcal{S} = \langle \sigma^1, \gamma_t^1, \gamma_b^1 \rangle \rightarrow \langle \sigma^2, \gamma_t^2, \gamma_b^2 \rangle \rightarrow \ldots$ be an infinite sequence of configurations generated by the transition system Σ. In this sequence, the transition $\langle \sigma^i, \gamma_t^i, \gamma_b^i \rangle \rightarrow \langle \sigma^{i+1}, \gamma_t^{i+1}, \gamma_b^{i+1} \rangle$ is derived by the application of a transition rule of Σ. The sequence $\mathcal{S}$ is called a* trace *of Σ. The trace $\mathcal{S}$ of Σ is called a* fair trace *if it is generated by a* fair run *of the transition system, that is, a run in which every transition rule that is enabled infinitely often, is also applied infinitely often.*

Note that every finite trace is a fair trace. Furthermore, the priority assumption does not affect fairness, since in any given configuration, there is only a finite number of goal-related transitions that can be applied. In the following, we consider only infinite traces (which can be guaranteed by adding an "idling" rule that transitions a configuration to itself if no other transition rules are applicable). We use σ^i, γ_b^i, and γ_t^i to denote the ingredients of the ith state of a trace $\mathcal{S}$.

Definition 3 *Let $\mathcal{S}$ be a* trace *of Σ. We define $B(\mathcal{S})$, called the belief trace of $\mathcal{S}$, to be the LTL-trace $s = s^1 s^2 \ldots$ (where s^i is a state assigning a truth value to each atomic proposition), if the following condition holds:*

$$\forall \phi \in P, \forall i \in \mathbb{N} : \ \sigma^i \models_{cwa} \phi \ \Leftrightarrow s^i \models \phi$$

Furthermore, observe that since ϕ is non-temporal, we also have that $s^i \models \phi \Leftrightarrow B(\mathcal{S}), i \models_{LTL} \phi$.

The following proposition states that for every trace there is one unique belief trace possible.

Proposition 1 *Let $\mathcal{S}$ be a trace of Σ. The belief trace $B(\mathcal{S})$ is uniquely determined by $\mathcal{S}$.*
Proof: *This proposition is the direct consequence of matching belief bases σ^i with states s^i using the closed-world assumption. I.e., state s^i assigns the truth value of propositions based on their truth values in σ^i using closed-world assumption.*

Assumption 1 *The achieve goal $A(\phi)$ is properly operationalized by a transition system S if the following condition holds for every trace $\mathcal{S}$ of Σ:*

$$\forall i : (A(\phi), active, \Delta) \in \gamma_b^i \ \Rightarrow \ \exists j \geq i : \sigma^j \models_{cwa} \phi$$

Assumption 2 *The maintain goal $M(\phi, \tau)$ is properly operationalized by a transition system S if the following condition holds for every trace $\mathcal{S}$ of Σ:*

$$\begin{aligned} \forall i : \ & (M(\phi, \tau), active, \Delta) \in \gamma_b^i \Rightarrow \\ & (\exists j \geq i : \ \sigma^j \models_{cwa} \tau) \ \wedge \ (\forall \, i \leq k < j : \ \sigma^k \models_{cwa} \phi) \\ \text{or} \quad & \tau = \bot \wedge (\forall k \geq i : \ \sigma^k \models_{cwa} \phi) \end{aligned}$$

The following propositions show that the operational semantics defined in section 5 correctly operationalize complex temporal goals (in γ_t) using the basic achievement and maintenance goals (in γ_b). Generally, correct realisation is the property that if a temporal LTL formula χ is in the temporal goal base of state i and is active, formally $(\chi, active, \Delta) \in \gamma_t^i$, then $B(\mathcal{S}), i \models_{LTL} \chi$. However, since this only holds for the particular goal patterns that have been operationalized, we prove this for each case separately. All propositions are based on assumptions 1 and 2.

Proposition 2 *If $(\Diamond \phi, active, \Delta) \in \gamma_t^i$, then $B(\mathcal{S}), i \models_{LTL} \Diamond \phi$ for all fair traces $\mathcal{S}$ of Σ.*
Proof: *Case 1: Assume that $\sigma^i \not\models_{cwa} \phi$. By the definition of the condition-action pairs for $\Diamond \phi$ and the transition rule for Drop, we have that, since $\Diamond \phi \in \gamma_t^i$, we must have that $A(\phi) \in \gamma_b^{i+1}$. Therefore by assumption 1 (and definition 3) we have that $\exists j \geq i + 1 : B(\mathcal{S}), j \models_{LTL} \phi$ and hence by the semantics of LTL that $B(\mathcal{S}), i \models_{LTL} \Diamond \phi$. Case 2: Assume that $\sigma^i \models_{cwa} \phi$. Hence $B(\mathcal{S}), i \models_{LTL} \phi$ and trivially $B(\mathcal{S}), i \models_{LTL} \Diamond \phi$.*

The following proposition states that a property can be maintained if it already holds.

Proposition 3 *If $(\Box \phi, active, \Delta) \in \gamma_t^i$ and $\sigma^i \models_{cwa} \phi$, then $B(\mathcal{S}), i \models_{LTL} \Box \phi$ for all fair traces $\mathcal{S}$ of Σ.*
Proof: *Since $\Box \phi \in \gamma_t^i$, by the definition of the condition-action rules and the transition rule for Drop, we have that $M(\phi, \bot) \in \gamma_b^{i+1}$. By assumption 2 (and definition 3), since $\sigma^i \models_{cwa} \phi$ and $\forall j \geq i + 1 : \sigma^j \models_{cwa} \phi$, we have $\forall j \geq i : B(\mathcal{S}), j \models_{LTL} \phi$ and hence $B(\mathcal{S}), i \models_{LTL} \Box \phi$.*

The following proposition shows that $\phi \cup \tau$ is correctly operationalized. It assumes that $\tau \neq \bot$, since $\phi \cup \bot$ is false in LTL.

Proposition 4 *If $(\phi \cup \tau, active, \Delta) \in \gamma_t^i$ (where $\tau \neq \bot$), and $\sigma^i \models_{cwa} \phi$, then $B(\mathcal{S}), i \models_{LTL} \phi \cup \tau$ for all fair traces $\mathcal{S}$ of Σ.*
Proof (sketch): *Since $(\phi \cup \tau) \in \gamma_t^i$, by the definition of the condition-action pairs and of the transition rules, we have $M(\phi, \tau) \in \gamma_b^{i+1}$. By assumption 2 the trace $\mathcal{S}$ at configuration $i + 1$ satisfies $\exists j \geq i + 1 : \sigma^j \models_{cwa} \tau \wedge \forall k : i + 1 \leq k < j : \sigma^k \models_{cwa} \phi$. From this condition together with the definition 3 and the fact that $B(\mathcal{S}), i \models_{LTL} \phi$, it is easy to see that $B(\mathcal{S}), i \models_{LTL} \phi \cup \tau$.*

Proposition 5 *If $(\Diamond(\tau \wedge \Box \phi), active, \Delta) \in \gamma_t^i$, then $B(\mathcal{S}), i \models_{LTL} \Diamond(\tau \wedge \Box \phi)$ for all fair traces $\mathcal{S}$ of Σ.*
Proof (sketch): *We consider two cases in configuration i. Case 1: $\sigma^i \not\models_{cwa} \phi \wedge \tau$ (either ϕ or τ is not believed in the current configuration). Since $\Diamond(\tau \wedge \Box \phi) \in \gamma_t^i$, by the definition of the condition-action pairs and of the transition rules, we have $A(\phi \wedge \tau) \in \gamma_b^{i+1}$. By assumption 1 (and definition 3), we have that $\exists j \geq i + 1 : B(\mathcal{S}), j \models_{LTL} \phi \wedge \tau$. Since $\Diamond(\tau \wedge \Box \phi)$ is never removed from the temporal goal base (the condition-action pairs always suspend, and never drop it), we have $\Diamond(\tau \wedge \Box \phi) \in \gamma_t^j$. By the definition of the condition-action pairs and of the transition rules, and because $\sigma^j \models_{cwa} \phi \wedge \tau$, we have $M(\phi, \bot) \in \gamma_b^{j+1}$. From, $\sigma^j \models_{cwa} \phi$, assumption 2 and the definition 3, we have $B(\mathcal{S}), j \models_{LTL} \Box \phi$ and hence $B(\mathcal{S}), i \models_{LTL} \Diamond(\tau \wedge \Box \phi)$. Case 2: $\sigma^i \models_{cwa} \phi \wedge \tau$. It is easy to see that the proposition holds in this case by having $i = j$ in the proof sketch of case 1.*

Proposition 6 *If $(\Diamond(\tau \wedge (\phi \cup \tau')), active, \Delta) \in \gamma_t^i$, then $B(\mathcal{S}), i \models_{LTL} \Diamond(\tau \wedge (\phi \cup \tau'))$ for all fair traces $\mathcal{S}$ of Σ.*
Proof (sketch): *The proof is similar to the proof of the previous proposition.*

The following proposition assumes that $\tau \rightarrow \phi$. The justification for this assumption is that we want to show that the temporal goal is correctly realised. In the case where τ becomes true but ϕ does not hold, the temporal goal immediately fails, i.e. *no* operationalisation is able to realise the goal. We thus exclude this case.

Proposition 7 *If $(\Box(\tau \rightarrow (\phi \cup \tau')), active, \Delta) \in \gamma_t^i$ and $\tau \rightarrow \phi$, then $B(\mathcal{S}), i \models_{LTL} \Box(\tau \rightarrow (\phi \cup \tau'))$ for all fair traces $\mathcal{S}$ of Σ.*
Proof (sketch): *We want to show that $B(\mathcal{S}), i \models_{LTL} \Box(\tau \rightarrow (\phi \cup \tau'))$, i.e. that for any $j \geq i$, we have $B(\mathcal{S}), j \models_{LTL} (\tau \rightarrow (\phi \cup \tau'))$. There are two cases. Case 1: $\sigma^j \not\models_{cwa} \tau$. In this case the implication trivially holds. Case 2: $\sigma^j \models_{cwa} \tau$. Since $\tau \rightarrow \phi$, we have $\sigma^j \models_{cwa} \phi$. Since the temporal goal is always suspended and never dropped, $\Box(\tau \rightarrow (\phi \cup \tau')) \in \gamma_t^j$ for all $j \geq i$. Thus, by the definition of*

the condition-action pairs and of the transition rules, and since $\sigma^j \models_{cwa} \tau \wedge \phi$*, we have* $M(\phi, \tau') \in \gamma_b^{j+1}$*. By assumption 2 the trace* $\mathcal{S}$ *at configuration* $j+1$ *ensures that* ϕ *will be entailed by the belief base until* τ' *is entailed. Because* $\sigma^j \models_{cwa} \phi$*, trace* $\mathcal{S}$ *at configuration* j *ensures that* ϕ *is entailed by the belief base until* τ' *is entailed, i.e.* $B(\mathcal{S}), j \models_{LTL} \phi \mathcal{U} \tau'$*, from which* $B(\mathcal{S}), j \models_{LTL} (\tau \rightarrow (\phi \mathcal{U} \tau'))$ *trivially follows.*

6. DISCUSSION

In this paper we built on a temporal logic view of agent goals by defining new, novel, goal types, and showing how these new goal types could be operationalized in terms of existing base goal types (achievement and maintenance). The operationalization is based on a goal life cycle where transitions between states of goals are governed by condition-action pairs. We show that the operationalization realizes traces on which the temporal goals are satisfied, assuming satisfaction of the basic goal types. Through this we have provided a flexible framework for operationalizing rich goal types. Other goal types can be added by providing their translation to the basic goal types. Although we have provided several examples in the paper, future work will have to show which goal types are particularly useful in practice. This may also depend on the domain that is modelled.

An important topic for future work to allow gaining more practical experience with the framework is implementing it on top of conventional agent-oriented programming languages, such as 2APL [6], Jadex [15], Jason [3], JACK [5] etc. It is interesting to note that the proposed framework appears to be a good match with rule-based systems, as used in Opal [20]. Also, we aim to extend the framework to include subgoals along the lines of [14], and then using this as a basis for incorporating goal suspension/resumption and abortion (based on [17, 18]).

7. REFERENCES

[1] F. Bacchus and F. Kabanza. Planning for temporally extended goals. In *Proceedings of the 13th National Conference on Artificial Intelligence (AAAI'96)*, pages 1215–1222, 1996.

[2] C. Baral and J. Zhao. Non-monotonic temporal logics for goal specification. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 236–242, 2007.

[3] R. H. Bordini, J. F. Hübner, and M. Wooldridge. *Programming multi-agent systems in AgentSpeak using Jason*. Wiley, 2007. ISBN 0470029005.

[4] L. Braubach and A. Pokahr. Representing long-term and interest BDI goals. In *Programming Multi-Agent Systems (ProMAS)*, 2009.

[5] P. Busetta, R. Rönnquist, A. Hodgson, and A. Lucas. JACK Intelligent Agents - Components for Intelligent Agents in Java. Technical report, Agent Oriented Software Pty. Ltd, Melbourne, Australia, 1998. Available from *http://www.agent-software.com*.

[6] M. Dastani. 2APL: a practical agent programming language. *Autonomous Agents and Multi-Agent Systems*, 16(3):214–248, 2008.

[7] M. Dastani, M. B. van Riemsdijk, and J.-J. Ch. Meyer. Goal types in agent programming. In *Proceedings of the 17th European Conference on Artifical Intelligence 2006 (ECAI'06)*, volume 141 of *Frontiers in Artificial Intelligence and Applications*, pages 220–224. IOS Press, 2006.

[8] S. Duff, J. Harland, and J. Thangarajah. On proactivity and maintenance goals. In *Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 1033–1040, Hakodate, 2006.

[9] E. Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B: Formal Models and Semantics, pages 996–1072. Elsevier, Amsterdam, 1990.

[10] M. Fisher and A. Hepple. Executing logical agent specifications. In R. H. Bordini, M. Dastani, J. Dix, and A. El Fallah-Seghrouchni, editors, *Multi-Agent Programming: Languages, Platforms and Applications*, volume 2, chapter 1, pages 3–29. Springer, 2009.

[11] K. Hindriks and M. B. van Riemsdijk. Satisfying maintenance goals. In *Declarative Agent Languages and Technologies (DALT'07)*, volume 4897 of *LNAI*, pages 86–103. Springer, 2008.

[12] K. V. Hindriks, W. van der Hoek, and M. B. van Riemsdijk. Agent programming with temporally extended goals. In *Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 137–144. IFAAMAS, 2009.

[13] S. M. Khan and Y. Lespérance. A logical account of prioritized goals and their dynamics. In G. Lakemeyer, L. Morgenstern, and M. A. Williams, editors, *Proc. of the 9th International Symposium on Logical Formalizations of Commonsense Reasoning*, pages 85–90, 2009.

[14] M. Morandini, L. Penserini, and A. Perini. Operational semantics of goal models in adaptive agents. In C. Sierra, C. Castelfranchi, K. S. Decker, and J. S. Sichman, editors, *Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 129–136. IFAAMAS, 2009.

[15] A. Pokahr, L. Braubach, and W. Lamersdorf. Jadex: a BDI reasoning engine. In R. H. Bordini, M. Dastani, J. Dix, and A. El Fallah Seghrouchni, editors, *Multi-Agent Programming: Languages, Platforms and Applications*. Springer, Berlin, 2005.

[16] S. Shapiro and G. Brewka. Dynamic interactions between goals and beliefs. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2625–2630, 2007.

[17] J. Thangarajah, J. Harland, D. Morley, and N. Yorke-Smith. Aborting goals and plans in BDI agents. In *Autonomous Agents and MultiAgent Systems (AAMAS)*, 2007.

[18] J. Thangarajah, J. Harland, D. Morley, and N. Yorke-Smith. Suspending and resuming tasks in intelligent agents. In Padgham, Parkes, Müller, and Parsons, editors, *Autonomous Agents and Multi-Agent Systems (AAMAS)*, 2008.

[19] M. B. van Riemsdijk, M. Dastani, and M. Winikoff. Goals in agent systems: A unifying framework. In Padgham, Parkes, Müller, and Parsons, editors, *Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 713–720. IFAAMAS, 2008.

[20] M. Wang, M. Nowostawski, and M. K. Purvis. Declarative agent programming support for a FIPA-compliant agent platform. In R. H. Bordini, M. Dastani, J. Dix, and A. El Fallah-Seghrouchni, editors, *ProMAS*, volume 3862 of *LNCS*, pages 252–266. Springer, 2005.

[21] M. Winikoff, L. Padgham, J. Harland, and J. Thangarajah. Declarative & procedural goals in intelligent agent systems. In *Proceedings of the Eighth International Conference on Principles of Knowledge Representation and Reasoning (KR)*, Toulouse, France, Apr. 2002.