Compiling Control Knowledge into Preconditions for Planning in the Situation Calculus

Alfredo Gabaldon
Department of Computer Science
University of Toronto
alfredo@cs.toronto.edu

Abstract

A promising technique used in some planning systems to improve their performance is the use of domain dependent search control knowledge. We present a procedure for compiling search control knowledge, expressed declaratively in a logic, into the preconditions of the plan actions (operators). We do this within the framework of the situation calculus by introducing a transformation of non-Markovian action theories into classical Markovian situation calculus theories.

1 Introduction

One of the strategies used in planning to mitigate the complexity of the general problem is to employ some form of domain specific knowledge to assist the search for a plan. For instance, Hierarchical Task Network (HTN) planning systems [Sacerdoti, 1974; Wilkins, 1988; Erol *et al,* 1996] use domain specific procedural knowledge in the form of task decomposition methods. The forward-chaining planners TLPlan [Bacchus and Kabanza, 2000], TALPlanner [Kvarnstrom and Doherty, 2000] and the SAT based planner SATPlan [Kautz and Selman, 1998] use domain knowledge in the form of declarative constraints expressed in a logical language. This strategy has been shown experimentally to yield remarkable improvements in performance. Both TLPlan and TALPlanner use control constraints expressed in the form of temporal logic formulas. These formulas are used to eliminate plan prefixes which will lead to a suboptimal plan or cannot be extended into a complete plan at all.

Most practical planning systems use STRIPS, ADL or extensions of these to describe actions and their effects. However, the first formal specification of the classical planning problem, due to [Green, 1969], was postulated in the language of the situation calculus [McCarthy, 1963]. The situation calculus has proven to be a very powerful formalism and has been employed in the formalization of many different aspects of dynamical systems (see e.g. [Reiter, 2001]). In this paper, we use the situation calculus as our formal framework and consider how to incorporate search control into action theories. Specifically, we show that control formulas with a certain syntactic form can be incorporated into nonMarkovian action theories in the situation calculus, as recently introduced in [Gabaldon, 2002], in a similar way as [Lin and Reiter, 1994] treat qualification state constraints. In these nonMarkovian action theories, the effects and preconditions of actions are not assumed to depend solely on the current situation, but on any past situation. This nonMarkovian property of the theories allows an easier incorporation of dynamic constraints into database specifications, a more natural and concise axiomatization of nonMarkovian operations like *rollback* in transaction systems and other domains with nonMarkovian properties. In this paper, we are concerned with another problem where these theories are useful: incorporating search control knowledge by treating it as dynamic, qualification state constraints. Dynamic in the sense that they may refer to the current and any past situation, as opposed to static state constraints which refer only to the current state. Moreover, they are understood as qualification constraints because they pose further restrictions on the "executability" of actions. (We do not consider the use of these constraints for expressing ramifications, i.e. indirect effects of actions.)

Furthermore, we present a transformation from nonMarkovian action theories into Markovian ones. This transformation takes an action theory with nonMarkovian axioms and by applying regression steps and introducing additional fluents and their corresponding successor state axioms, produces a classical Markovian theory as introduced by [Reiter, 1991 J. We then show how this transformation can be used for compiling search control knowledge into normal action preconditions. This nonMarkovian to Markovian transformation procedure is of independent interest. Toward the end we extend this approach for theories with explicit time and close with a discussion.

2 Formal Preliminaries

In this section we give an overview of the situation calculus and the main definitions necessary for action theories without the Markov assumption.

2.1 The Situation Calculus

The situation calculus is a first order logic language with three basic components: actions, situations, and fluents. Actions are responsible for all the changes in the world. Situations are sequences of actions which represent possible histories of the world. Fluents are properties of the world that

change from situation to situation as a result of the execution of actions. Formally, the language has three sorts: *action, situation* and *fluent.* In addition to variables of these sorts, the language includes functions such as $move(x, y)$ to represent actions, a constant S_0 and a function $do(a, s)$ for situations such as $do(move(x, y), S_0)$, and predicates for representing fluents such as $atLoccition(x, l, do(a, s))$. The initial situation, or empty history, is denoted by the constant *So-* Non-empty histories are built by means of the function *do.* For a complete description see [Pirri and Reiter, 1999; Reiter,2001].

A situation calculus axiomatization of a domain, includes the following set of axioms:

1. For each action function $A(\vec{x})$, an *action precondition axiom* of the form: $Poss(A(x_1, \ldots, x_n), s) \equiv \Pi_A(x_1, \ldots, x_n, s)$ where s is the only term of sort situation in $\Pi_A(x_1, \ldots, x_n, s)$.
2. For each fluent $F(\vec{x}, s)$, a *successor state axiom* of the form: $F(x_1, \ldots, x_n, do(a, s)) \equiv \Phi_F(x_1, \ldots, x_n, a, s)$ where s is the only term of sort situation in $\Phi_F(x_1, \ldots, x_n, a, s)$.
3. Unique names axioms for actions. For instance: $move(x, y) \neq pickup(x)$.
4. Axioms describing the initial situation of the world: a finite set of sentences whose only situation term is the constant *So.*

A set of these axioms, together with a set of domain independent foundational axioms Σ, is called a (Markovian) *basic action theory.*

2.2 A nonMarkovian Situation Calculus

For a basic action theory without the Markov assumption, we need some definitions. These are based on those in [Gabaldon, 2002].

Wc will use the following abbreviations in the definitions to follow:

$$(\exists s : \sigma' \sqsubset \sigma'' \sim \sigma)W \stackrel{\text{def}}{=} (\exists s)[\sigma' \sqsubset \sigma'' \wedge \sigma'' \sim \sigma \wedge W]$$
$$(\forall s : \sigma' \sqsubset \sigma'' \sim \sigma)W \stackrel{\text{def}}{=} (\forall s)[(\sigma' \sqsubset \sigma'' \wedge \sigma'' \sim \sigma) \supset W]$$

where $\sim$ stands for either $\sqsubset$ or $=$ and variable .s appears in σ''. If σ' is S_0 then we may write $(\exists s : \sigma'' \sim \sigma)W$ and $(\forall s : \sigma'' \sim \sigma)W$ instead.

Definition 1 (Bounded Formulas) For $n \geq 0$, let σ be a term $do([\alpha_1, \ldots, \alpha_n], \lambda)$ rooted at λ. The formulas of $\mathcal{L}_{situalc}$ *bounded* by *o* are the smallest set of formulas such that:

1. If *W* is an atom whose situation terms are all rooted at λ, then *W* is bounded by σ.
2. If *W, W"* are formulas bounded by situation terms rooted at *s* and λ, respectively, then $(\exists s : \sigma' \sqsubset \sigma'' \sim \sigma)W$ and $(\forall s : \sigma' \sqsubset \sigma'' \sim \sigma)W$ are formulas bounded by σ, where σ'' is rooted at *s* and $W = (\neg)(W' \wedge W'')$.
3. If Wi,W2 are formulas bounded by situation terms rooted at λ, then $\neg W_1$, $W_1 \wedge W_2$ and $(\exists v)W_1$, where v is of sort *action* or *object,* are formulas bounded by σ.

The set of formulas *strictly bounded* by σ is similarly defined by requiring in item (1) above that all situation terms of *W* be subterms of σ, in item (2) that *W* be strictly bounded by a subterm of σ'' and *W"* by a subterm of rr; and in item (3) that *W\, W2* be strictly bounded by subterms of σ.

Example 1 Past temporal logic connectives can be expressed in the situation calculus with strictly bounded formulas as follows:

1. $prev(\varphi, s) \stackrel{\text{def}}{=} (\exists a).(\exists s' : do(a, s') = s)\ \varphi(a, s')]$.
2. $since(\varphi_1, \varphi_2, s) \stackrel{\text{def}}{=} (\exists s' : s' \sqsubset s).\ \varphi_2(s') \wedge (\forall s'' : s' \sqsubset s'' \sqsubseteq s)\ \varphi_1(s'')$.
3. $sometime(\varphi, s) \stackrel{\text{def}}{=} (\exists s' : s' \sqsubset s)\ \varphi(s')$.
4. $always(\varphi, s) \stackrel{\text{def}}{=} (\forall s' : s' \sqsubset s)\ \varphi(s')$.

NonMarkovian basic action theories differ from those which include the Markov assumption in that preconditions and effects of actions may depend on any past situation, not only on the current one.

Hence the rhs, $\Pi_A(x_1, \ldots, x_n, s)$, of action precondition axioms in a nonMarkovian basic action theory are formulas bounded by situation term *s* which do not mention predicate *Poss* and may refer to past situations. Similarly, the rhs, $\Phi_F(x_1, \ldots, x_n, a, s)$ of successor state axioms are formulas strictly bounded by .s.

2.3 Regression

For basic action theories with the Markov assumption, [Pirri and Reiter, 1999] define a provenly correct *regression* mechanism that takes a situation calculus sentence and, under certain restrictions on the form of this sentence, transforms it into an equivalent sentence whose only situation term is 5(>. This allows proving sentences without appealing to the foundational axioms Σ which include a second order axiom. This regression operator was generalized for nonMarkovian theories in [Gabaldon, 2002].

In a nutshell, the regression operator, denoted by $\mathcal{R}$, takes a sentence and recursively replaces each fluent atom $F(\vec{t}, do(\alpha, \sigma))$ by its definition according to its successor state axiom, i.e. by $\Phi_F(\vec{t}, \alpha, \sigma)$. Atoms $Poss(A(\vec{t}), do(\alpha, \sigma))$ are similarly replaced by their definitions given by the action precondition axioms. Regression recursively replaces these atoms until all the situation terms are reduced to So- For lack of space we refer the reader to [Pirri and Reiter, 1999; Reiter, 2001; Gabaldon, 2002] for the formal details.

In the transformation operator we introduce later, we will use a one-step version of the regression operator: $TV[W[do(\alpha, \sigma)]]$ stands for the regression of a sentence $W[do(\alpha, \sigma)]$ bounded by $do(\alpha, \sigma)$ that results in a sentence bounded by *o.*

3 Control Knowledge and the Qualification Problem

As mentioned earlier, the use of search control knowledge has proven to be a promising recourse for improving the performance of planning systems. Both TLPlan [Bacchus and

Kabanza, 2000] and TALPlanner [Kvarnstrom and Doherty, 2000] use declarative control knowledge in the form of temporal logic and have shown substantial computational improvement.

Since our goal is to compile search control into preconditions, we express control knowledge in terms of the past, not the future as it is done in TLPlan and TALPlanner (this is further discussed in the last section). However, declarative search control knowledge is typically expressed in a future temporal logic and we believe that obtaining preconditions from knowledge in that form is an important problem. In the last section we comment on how we are approaching this.

From the point of view of logical theories of action, such as the situation calculus axiomatizations we discuss in this paper, taking control knowledge into account is closely related to the classical *qualification problem* [McCarthy, 1977]. This is the problem of determining all the conditions that need to be satisfied for an action to be executable and control knowledge is effectively a set of additional constraints on the cxecutability of actions, i.e. on what actions can be considered part of an executable plan.

Lin & Reiter [1994] have considered the qualification problem for situation calculus basic action theories with *state constraints,* which are formulas of the form $(\forall s)Q(s)$, uniform in s. Their solution is to add the qualifications to the precondition axiom of each action type:

$$Poss(A(\vec{x}), s) \equiv \Pi_A(\vec{x}, s) \wedge \Pi_Q,$$

where $\Pi_Q \stackrel{\text{def}}{=} [Q(do(A(\vec{x}), s))]$. After adding the qualifications to the precondition axioms, one can discard the constraints if two conditions are satisfied: all the state constraints hold initially, i.e. *Q(So)* holds for each *Q*\ and the domain closure assumption on actions is included in the theory.

Control formulas are similar to state constraints in that they pose additional qualifications on actions, but they are more general since they may quantify over past situations.

In the nonMarkovian situation calculus, successor state axioms and action precondition axioms can refer to past situations in addition to the current situation. This permits control knowledge to be incorporated into action precondition axioms in the same way Lin & Reiter do. Formally, if *C(s)* is a formula bounded by .s representing some piece of control knowledge, we take this into account by adding it as an additional precondition for actions:

$$Poss(A(\vec{x}), s) \equiv \Pi_A(\vec{x}, s) \wedge C(do(A(\vec{x}), s)). \quad (1)$$

Precondition axioms in nonMarkovian action theories allow situation terms rooted at *s* such as $do(A(\vec{x}), s)$ to appear in their rhs, so we can add $C(do(A(\vec{x}), s))$ directly without modification. By including the domain closure assumption on actions, we are then guaranteed that a situation is executable only if it satisfies the constraints.

Example 2 *(Logistics domain constraint)* Consider a constraint from a logistics domain, which involves vehicles delivering packages to locations, saying that a truck must remain in a location if there is a package at that location and it's never been in the truck (so it hasn't just been delivered and needs to be moved):

$$(\forall p, t, loc).$$
$$since(\neg inTruck(p, t), atObj(p, loc) \wedge atTruck(t, loc), s)$$
$$\supset$$
$$since(atTruck(t, loc), atObj(p, loc) \wedge atTruck(t, loc), s)$$

Suppose that the action precondition axiom for $driveTruck(t, loc)$, driving a truck t to a location *loc* is:

$$Poss(driveTruck(t, loc), s) \equiv$$
$$(\exists loc').atLoc(t, loc', s) \wedge loc' \neq loc.$$

We can add the control formula as a new conjunct in the rhs after simply replacing *$* with $s^* = do(driveTruck(t, loc), s)$:

$$Poss(driveTruck(t, loc), s) \equiv$$
$$(\exists loc').atLoc(t, loc', s) \wedge loc' \neq loc \wedge (\forall p, t, loc).$$
$$since(\neg inTruck(p, t), atObj(p, loc) \wedge atTruck(t, loc), s^*)$$
$$\supset$$
$$since(atTruck(t, loc), atObj(p, loc) \wedge atTruck(t, loc), s^*)$$

In the next section, we introduce an operator which takes a basic action theory with such nonMarkovian axioms, and transforms it into one with Markovian ones.

4 Transforming a nonMarkovian Theory into Markovian

In order simplify the presentation, let us first make some simplifying assumptions on the form and the nesting of formulas on which we define the transformation.

Formally, in reference to item (2) of Definition 1, we will assume that formulas are of the forms:

$$(\exists s' : s^* \sqsubset s' \sqsubset s)\ (W_1[s'] \wedge (\forall s'' : s' \sqsubset s'' \sqsubseteq s) W_2[s'']) \quad (2)$$

$$(\forall s' : s^* \sqsubset s' \sqsubset s)\ (W_1[s'] \wedge (\exists s'' : s' \sqsubset s'' \sqsubseteq s) W_2[s'']) \quad (3)$$

where all (if any) free variables of W2 are among the free variables of the full formula. Other combinations of quantifiers and logical connectives reduce to these two cases. The restriction on the free variables of W2 does make the transformation simpler. We treat the general case in the full version of this paper. Next, we restrict the nesting of formulas in the following way. By definition, *W2* must itself be of the form $(\neg)(W' \wedge W'')$ where W' is bounded by *s"* and \Y" by *s.* We will restrict *W-i* by requiring it to be of the form $(\neg)$ *W',* i.e. without the conjunct *W".*

Notice however that, even with this restrictions, arbitrary nesting of the past temporal logic abbreviations (Example 1) is still expressible.

The transformation performs a combination of regression and of replacement of formulas by new fluents:

Definition 2 *Let φ be an $\mathcal{L}_{situalc}$ formula bounded by s. We define a formula $\mathcal{M}[\varphi]$ obtained from φ recursively as follows. By Definition 1, formula φ must be of one of the following forms:*

1. *An atom whose only situation term is s. Then* $\mathcal{M}[\varphi] = \varphi$.
2. *A formula of the form (2). Then,* $\mathcal{M}[\varphi] = P_\varphi(\vec{x}, s)$ *which is a new fluent with successor state axiom:*

$$P_\varphi(\vec{x}, do(a,s)) \equiv \mathcal{R}^1[P_{W_2}(\vec{x}_2, do(a,s))] \wedge \{P_{W_1}(\vec{x}_1, s) \vee P_\varphi(\vec{x}, s)\}$$

where $\mathcal{M}[W_1] = P_{W_1}(\vec{x}_1, s)$ *and* $\mathcal{M}[W_2] = P_{W_2}(\vec{x}_2, s)$.
Initially, the new fluent is set to false, i.e. $(\forall \vec{x}) \neg P_\varphi(\vec{x}, S_0) \in \mathcal{D}_{S_0}$.
3. *A formula of the form (3). Then,* $\mathcal{M}[\varphi] = P_\varphi(\vec{x}, s)$ *which is a new fluent with successor state axiom:*

$$P_\varphi(\vec{x}, do(a,s)) \equiv P_{W_1}(\vec{x}_1, s) \wedge \{\mathcal{R}^1[P_{W_2}(\vec{x}_2, do(a,s))] \vee P_\varphi(\vec{x}, s)\}.$$

where $\mathcal{M}[W_1] = P_{W_1}(\vec{x}_1, s)$ *and* $\mathcal{M}[W_2] = P_{W_2}(\vec{x}_2, s)$ *as before, and the new fluent is initially set to true, i.e.* $(\forall \vec{x}) P_\varphi(\vec{x}, S_0) \in \mathcal{D}_{S_0}$.

Theorem 1 Let φ denote a formula of the form (2) or (3) and $\mathcal{D}_\varphi$ be the basic action theory obtained from $\mathcal{D}$ by adding the fluents and axioms resulting from applying $\mathcal{M}$ to φ. Then,

$$\mathcal{D}_\varphi \models (\forall)[\varphi \equiv P_\varphi(\vec{x}, s)].$$

Now, we can apply the transformation to a nonMarkovian action theory $\mathcal{D}$ whose action precondition axioms have as an additional condition a constraint *C(s),* and obtain a Markovian theory $\mathcal{D}^{\mathcal{M}}$ that enforces the constraint. This produces a theory with precondition axioms of the form:

$$Poss(A(\vec{x}), s) \equiv \Pi_A(\vec{x}, s) \wedge \mathcal{M}[C(do(A(\vec{x}), s))].$$

Theorem 2 Let *C(s)* be a control knowledge formula bounded by *s* and let $\mathcal{D}$ be a nonMarkovian basic action theory whose precondition axioms have the form (1) and includes domain closure axioms for actions. Then,

$$\mathcal{D}^{\mathcal{M}} \models (\forall a, s).Poss(a, s) \supset C(do(a, s)).$$

Example 3 Consider again the logistics domain control formula from Example 2. Let φ_1 and φ_2 stand for premise and consequence, respectively. Both formulas are of form (2). The premise φ_1 stands for:

$$(\exists s' : s' \sqsubset s)\{atObj(p, loc, s') \wedge atTruck(t, loc, s') \wedge (\forall s'' : s' \sqsubset s'' \sqsubseteq s) \neg inTruck(p, t, s'')\}$$

Applying operator $\mathcal{M}$ on φ_1 produces three new fluents $P_1(p,t,loc,s)$, $P_2(p,t,loc,s)$ and $P_{\varphi_1}(p,t,loc,s)$ with the following successor state axioms:

$$P_1(p,t,loc,do(a,s)) \equiv atObj(p,loc,s) \wedge atTruck(t,loc,s) \vee P_1(p,t,loc,s)$$

$$P_2(p,t,loc,do(a,s)) \equiv \mathcal{R}^1[\neg atTruck(t,loc,do(a,s))] \wedge \{P_2(t,loc,s) \vee (atObj(p,loc,s) \wedge atTruck(t,loc,s))\}$$

$$P_{\varphi_1}(p,t,loc,do(a,s)) \equiv \mathcal{R}^1[P_2(t,loc,do(a,s))] \wedge P_1(p,t,loc,s) \vee P_{\varphi_1}(p,t,loc,s)$$

The result of $\mathcal{M}$ on φ_2 is similar. Then the constraint becomes:

$$P_{\varphi_1}(p,t,loc,s) \supset P_{\varphi_2}(p,t,loc,s).$$

Finally, we add the constraint to action precondition axioms:

$$Poss(A(\vec{x}), s) \equiv \Pi_A(\vec{x}, s) \wedge \mathcal{R}^1[P_{\varphi_1}(p,t,loc,do(A(\vec{x}),s)) \supset P_{\varphi_2}(p,t,loc,do(A(\vec{x}),s))].$$

Example 4 *(Closed form solution for a fluent)* Consider a fluent *F(x, s)* and its closed form solution [Reiter, 2001]:

$$\begin{aligned} &F(\vec{x}, s) \equiv \\ &F(\vec{x}, S_0) \wedge \neg(\exists s_1 : s_1 \sqsubseteq s)\{(\exists a', s'). \\ &\qquad do(a', s') = s_1 \wedge \gamma_F^-(\vec{x}, a', s')\} \vee \\ &(\exists s_1 : s_1 \sqsubseteq s)\{(\exists a', s').do(a', s') = s_1 \wedge \gamma_F^+(\vec{x}, a', s') \wedge \\ &\qquad (\forall s_2 : s_1 \sqsubseteq s_2 \sqsubseteq s). \\ &\qquad \neg(\exists a'', s'').do(a'', s'') = s_2 \wedge \gamma_F^-(\vec{x}, a'', s'')\} \end{aligned} \tag{4}$$

Clearly, the rhs of the above sentence is formed by $F(\vec{x}, S_0)$ and two formulas which are almost of the form (2). The difference is in the $\sqsubseteq$'s immediately after the existential quantifiers on situations *s1,s2,* which would need to be strict $\sqsubset$ to fit form (2), and the subformulas of the form $(\exists a', s')do(a', s') = s$. The latter will be simplified away. Further, it is easy to modify the transformation on (2) to handle the $\sqsubseteq$'s. In general, all we need to do is replace $P_{W_1}(\vec{x}_1, s)$ with $\mathcal{R}^1[P_{W_1}(\vec{x}_1, do(a,s))]$. However, in this example the one step regression won't be necessary.

Applying the transformation to the subformula

$$\varphi_1 = (\exists s_1 : s_1 \sqsubseteq s)\{(\exists a', s').do(a', s') = s_1 \wedge \gamma_F^-(\vec{x}, a', s')\}$$

results in a predicate $P_{\varphi_1}(\vec{x}, s)$ with successor state axiom (after simplifying) $P_{\varphi_1}(\vec{x}, do(a,s)) \equiv \gamma_F^-(\vec{x}, a, s) \vee P_{\varphi_1}(\vec{x}, s)$
The transformation of subformula $\varphi_2 =$

$$(\exists s_1 : s_1 \sqsubseteq s)\{(\exists a', s').do(a', s') = s_1 \wedge \gamma_F^+(\vec{x}, a', s') \wedge (\forall s_2 : s_1 \sqsubseteq s_2 \sqsubseteq s). \{\neg(\exists a'', s'')do(a'', s'') = s_2 \wedge \gamma_F^-(\vec{x}, a'', s'')\}$$

results in a predicate $P_{\varphi_2}(\vec{x}, s)$ with successor state axiom (after simplifying) $P_{\varphi_2}(\vec{x}, do(a,s)) \equiv \gamma_F^+(\vec{x}, a, s) \vee \neg\gamma_F^-(\vec{x}, a, s) \wedge P_{\varphi_2}(\vec{x}, s)$
Noticeably, this successor state axiom for $P_{\varphi_2}(\vec{x}, s)$ is almost identical to the canonical form successor state axiom for $F(\vec{x}, s)$ which is $F(\vec{x}, do(a,s)) \equiv \gamma_F^+(\vec{x}, a, s) \vee F(\vec{x}, s) \wedge \neg\gamma_F^-(\vec{x}, a, s)$.

Finally, after applying the transformation to subformulas φ_1, φ_2 of the closed form formula (4), we obtain the following equivalent sentence:

$$F(\vec{x}, s) \equiv F(\vec{x}, S_0) \wedge \neg P_{\varphi_1}(\vec{x}, s) \vee P_{\varphi_2}(\vec{x}, s)$$

The transformation comes close to producing a canonical Markovian successor state axiom. It does not because the closed form solution formula uses information about the value of *F* in the initial situation.

5 Constraints with explicit time

In this section, we extend our approach to search control knowledge with explicit time. In the temporal situation calculus [Reiter, 2001], actions have an additional temporal argument denoting the time of occurrence, e.g. *drive(truck,loc,3.2)* for driving a *truck* to a location *loc* at time 3.2. There are also some additional axioms: $time(A(\vec{x},t)) = t$ specifying the occurrence time for each action A, and $start(do(a,s)) = time(a)$ specifying the start time of situations.

We will use $[t_1,t_2]+t$ to denote $[t_1+t, t_2+t]$ and $t \in [t_1,t_2]$ to denote $t_1 \leq t \wedge t \leq t_2$. So $start(s) \in [t_1,t_2] + start(s')$ stands for $start(s') + t_1 \leq start(s) \wedge start(s) \leq start(s') + t_2$ which means that s' precedes s by at least t_1 and at most t_2.

Analogous to forms (2) and (3), we have:

$$(\exists s' : s^* \sqsubset s' \sqsubset s) \{W_1[s'] \wedge start(s) \in [t_1,t_2] + start(s') \wedge (\forall s'' : s' \sqsubset s'' \sqsubseteq s) W_2[s'']\} \quad (5)$$

$$(\forall s' : s^* \sqsubset s' \sqsubset s) \{W_1[s'] \wedge start(s) \in [t_1,t_2] + start(s') \wedge (\exists s'' : s' \sqsubset s'' \sqsubseteq s) W_2[s'']\} \quad (6)$$

For a formula of the form (5), $\mathcal{M}$ yields a predicate $P_\varphi(\vec{x},t,s)$ with a successor state axiom:

$$P_\varphi(\vec{x},t,do(a,s)) \equiv \mathcal{R}^1[P_{W_2}(\vec{x}_2,do(a,s))] \wedge \{t = start(s) \wedge P_{W_1}(\vec{x}_1,s) \vee P_\varphi(\vec{x},t,s)\} \quad (7)$$

and $\mathcal{M}[(5)] = P_\varphi(\vec{x},t,s) \wedge start(s) \in [t_1,t_2]+t$. The initial database would include $(\forall t)\neg P_\varphi(\vec{x},t,S_0)$.

For a formula of the form (6), the successor state axiom for $P_\varphi(\vec{x},t,s)$ is:

$$P_\varphi(\vec{x},t,do(a,s)) \equiv P_{W_1}(\vec{x}_1,s) \wedge \{\mathcal{R}^1[P_{W_2}(\vec{x}_2,do(a,s))] \wedge t = start(s) \vee P_\varphi(\vec{x},t,s)\} \quad (8)$$

with $\mathcal{M}[(6)] = P_\varphi(\vec{x},t,s) \wedge start(s) \in [t_1,t_2] + t$. The initial database would include $(\forall t)P_\varphi(\vec{x},t,S_0)$.

The statement of the following theorem, which establishes the correctness of the transformation, requires adding the conjunct $t = start(s')$ to formulas (5) and (6):

$$(\exists s' : s^* \sqsubset s' \sqsubset s) \{W_1[s'] \wedge start(s) \in [t_1,t_2] + start(s') \wedge t = start(s') \wedge (\forall s'' : s' \sqsubset s'' \sqsubseteq s) W_2[s'']\}$$

Similarly for formula (6).

Theorem 3 Let φ denote a formula of the form (5) or (6) and define $\mathcal{D}^{\mathcal{M}}$ as before. Then,

$$\mathcal{D}^{\mathcal{M}} \models (\forall)[\varphi \equiv \{P_\varphi(\vec{x},t,s) \wedge start(s) \in [t_1,t_2] + t\}].$$

6 Discussion and conclusion

We have shown that incorporating control knowledge as additional preconditions in nonMarkovian action theories is trivial when this knowledge is in the form of bounded formulas (which include encodings of Past temporal logic modalities). We then introduced an operator for transforming non-Markovian basic action theories into Markovian ones. This operator introduces into a theory the additional fluents and axioms needed for keeping track of the relevant past information. We then showed how this operator can be used for compiling TLPlan style search control knowledge into action preconditions in the situation calculus.

In the TLPlan system, control knowledge is expressed in terms of linear (future) temporal logic, that is, temporal modalities *next, always in the future, until, and sometime in the future* are used. Every time a new operator is added to the plan prefix being considered, a control formula is *progressed* through it. If the formula is progressed into *false,* the plan prefix is pruned since a plan with this prefix will violate the constraint. Although it is easy to write situation calculus formulas corresponding to future temporal logic, regression cannot be used on them. However, we argue that if instead of progression, as in TLPlan, we evaluate the control formulas against plan prefixes, it is reasonable to use a logic that refers to past situations instead of future situations. Indeed, all the reasoning used by a planning system to decide if a partial plan should be discarded or not must be done relative to the properties of the partial plan itself. Furthermore, some future temporal formulas are satisfied by all plan prefixes (e.g. "sometime in the future ϕ" where ϕ is not unsatisfiable) and are not useful for search control. Thus it seems to us that restricting control formulas to refer exclusively to the past is appropriate.

Moreover, using past temporal logic is semantically cleaner for the following reason. The semantics of future temporal logic is defined in terms of infinite sequences of states. However, plans are finite sequences of actions and thus produce a finite sequence of states. In order to deal with this technical difficulty, TLPlan makes the assumption that the world never changes after the plan is completely executed and therefore the last state infinitely repeats. This assumption is reasonable under another assumption typically made in classical planning: that the agent executing the plan is the only agent that changes the world, so when this agent terminates executing its plan, the world remains unchanged. This assumption is unnecessary if using past temporal logic or bounded situation calculus formulas.

Nevertheless, it would be convenient to be able to handle control knowledge expressed in future temporal logic. We are currently working on developing a procedure that would allow us to take a future temporal logic control formula, such as this Briefcase domain one:

$$goal(at(x,loc)) \supset atBriefcase(loc) \textbf{ Until } \neg inBriefcase(x)$$

and produce a bounded situation calculus formula:

$$(\forall s_1 : S_0 \sqsubseteq s_1 \sqsubseteq s).\neg goal(at(x,loc),s_1) \vee (\forall a,s_2 : s_1 \sqsubseteq do(a,s_2) \sqsubseteq s)\{atBriefcase(loc,s_2) \vee (\exists s_3 : s_1 \sqsubset s_3 \sqsubset do(a,s_2))\neg inBriefcase(x,s_3)\}$$

that a plan prefix must satisfy. In turn, we can then apply the procedure introduced in this paper to compile these formulas into the preconditions of actions.

Bacchus & Kabanza [1998] have also extended their system for planning with *temporally extended goals,* which are conditions on sequences of states and not only on the final state as in classical planning. They use a first-order extension of the Metric Interval Temporal Logic (MITL) [Alur *et*

[1]Thc variable *t* is a fresh new variable so there is no effect on the meaning of the formulas.

ai, 1991] to specify the temporally extended goals. Defining plan correctness with respect to M1TL formulas as goals again requires one to make the assumption that the world remains unchanged after the plan is executed. In the case of temporally extended goals this is a further complication since testing if the current state satisfies the goal ϕ means checking whether ϕ is true if the world were to indefinitely remain as it currently is. (Past) MITL formulas can be encoded as formulas of the situation calculus with explicit time as discussed in the previous section. Hence our approach also shows a way to reduce planning with temporally extended goals into planning with classical goals.

Some preliminary experiments by Bacchus & Ady [1999] have shown that compiling search control knowledge into action preconditions can result in better performance compared to systems such as TLPlan which keep control knowledge as a separate set of formulas. A general method for compiling temporal formulas into preconditions in their framework has not yet been developed.

The work of [Rintanen, 2000] is also concerned with the use of control knowledge as additional preconditions of plan operators. He uses "auxiliary facts" which is a similar to our introduction of new fluents in the transformation. Like Bacchus & Kabanza, Rintanen uses future linear temporal logic for representing control knowledge for planers with ADL-like operators. With the help of these auxiliary facts, he then compiles the control knowledge into the operators. The main difference between his work and ours is that his framework is propositional, while ours is in the more expressive first order situation calculus. Furthermore, he does not allow nesting of temporal operators-in our framework this would correspond to disallowing nesting of quantifiers on situations-which we do allow. Also related but with a different approach, is the work of Sierra [1998]. There, domain knowledge in the form of *action selection rules* is used for controlling a STRIPS planner. Analyzing the relationship of Sierra's work with ours is among our future work.

References

[Alur etal., 1991] Rajeev Alur, Tomas Feder, and Thomas Henzinger. The benefits of relaxing punctuality. In *Proceedings of the Thenth Annual ACM Symposium on Princeples of Distributed Computing (PODC'91),* pages 139-152,1991.

[Bacchus and Ady, 1999] Fahiem Bacchus and Michael Ady. Precondition control. 1999.

[Bacchus and Kabanza, 1998] Fahiem Bacchus and Froduald Kabanza. Planning for temporally extended goals. *Annals of Mathematics and Artificial Intelligence,* 22:5-27, 1998.

[Bacchus and Kabanza, 2000] Fahiem Bacchus and Froduald Kabanza. Using temporal logics to express search control knowledge for planning. *Artificial Intelligence,* 116:123-191,2000.

[Erol *et al,* 1996] Kutluhan Erol, James A. Hendler, and Dana S. Nau. Complexity results for hierarchical task-network planning. *Annals of Mathematics and Artificial Intelligence,* 18:69-93, 1996.

[Gabaldon, 2002] Alfredo Gabaldon. Non-markovian control in the situation calculus. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence (AAAI'02),* pages 519-524, Edmonton, Canada, 2002.

[Green, 1969] C.C. Green. Theorem proving by resolution as a basis for question-answering systems. In B. Meltzer and D. Michie, editors, *Machine Intelligence 4,* pages 183-205. American Elsevier, New York, 1969.

[Kautz and Selman, 1998] Henry Kautz and Bart Selman. The role of domain-specific knowledge in the planning as satisfiability framework. In *Proceedings of the Fourth International Conference on Artificial Intelligence Planning Systems (AIPS'98),* pages 181-189,1998.

[Kvarnstrom and Doherty, 2000] Jonas Kvarnstrom and Patrick Doherty. TALplanner: A temporal logic based forward chaining planner. *Annals of Mathematics and Artificial Intelligence,* 30:119-169,2000.

[Lin and Reiter, 1994] Fangzen Lin and Ray Reiter. State constraints revisited. *Journal of Logic and Computation,* 4(5):655-678, October 1994.

[McCarthy, 1963] J. McCarthy. Situations, actions and causal laws. Technical report, Stanford University, 1963. Reprinted in Semantic Information Processing (M. Minsky ed.), MIT Press, Cambridge, Mass., 1968, pp. 410-417.

[McCarthy, 1977] John McCarthy. Epistemological problems of artificial intelligence. In *Proceedings of the Fifth International Conference on Artificial Intelligence (IJCAI77),pages* 1038-1044,1977.

[Pirri and Reiter, 1999] Fiora Pirn and Ray Reiter. Some contributions to the metatheory of the Situation Calculus. *Journal of the ACM,* 46(3):325-364,1999.

[Reiter, 1991] R. Reiter. The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result for goal regression. In V. Lifschitz, editor, *Artificial Intelligence and Mathematical Theory of Computation,* pages 359-380. Academic Press, 1991.

[Reiter, 2001] Ray Reiter. *Knowledge in Action: Logical Foundations for Describing and Implementing Dynamical Systems.* MIT Press, Cambridge, MA, 2001.

[Rintanen, 2000] Jussi Rintanen. Incorporation of temporal logic control into plan operators. In W. Horn, editor, *Procs. of the 14th European Conference on Artificial Intelligence (ECAI'00),* pages 526-530, Amsterdam, 2000. IOS Press.

[Sacerdoti, 1974] E.D. Sacerdoti. Planning in a hierarchy of abstraction spaces. *Artificial Intelligence,* 5:115-135,

[Sierra, 1998] Josefina Sierra. Declarative formalization of strategies for action selection. In *Procs. of the 7th Intl. Workshop on Nonmonotonic Reasoning NMR '98,* 1998.

[Wilkins, 1988] David E. Wilkins. *Practical Planning: Extending the classic AI planning paradigm.* Morgan Kaufmann, San Mateo, CA, 1988.