An Ordering on Goals for Planning – Formalizing Control Information in the Situation Calculus

Fangzhen Lin
6 King's College Road
Department of Computer Science
University of Toronto
Toronto, Canada M5S 1A4
fl@cs.toronto.edu

Abstract

In the situation calculus, plans and legal situations are isomorphic: a situation is legal iff it is the result of executing a plan in the initial situation. This isomorphism is important. It provides a basis on which certain control information in planning can be formalized in the situation calculus. As an application of this isomorphism, we investigate thoroughly a notion of goal ordering, and show its usefulness by applying it in a provably correct way to both linear and nonlinear planning.

1 Introduction

One of the features of the situation calculus (McCarthy [14]) is that a situation carries information about its history. For example, from looking at the syntax of the situation $do(move(B), do(move(A), S_0))$, we know that it is the result of executing $move(B)$ after $move(A)$ in the situation S_0. Conversely, given any situation, we obtain a new situation by executing a sequential plan in it. This natural isomorphism between situations and plans makes the situation calculus ideal for expressing certain knowledge that requires quantifications over plans, because the quantifications can then be replaced by quantifications over situations, which are first-order objects in the situation calculus.

We argue that this isomorphism is particularly useful to planning in that it facilitates the formalization of control information, which frequently involves quantifications over plans. As an example, this paper considers in detail the issue of goal ordering in classical planning.

In classical planning (Fikes and Nilsson [4]), a goal consists of a set of subgoals. In searching for a plan, most planners assume by default that subgoals do not interact. When they do interact, however, the order in which they are solved may be important.

To address this problem, Korf [9] proposes a taxonomy of subgoal interactions. According to Korf, a set of subgoals can be either independent, serializable, or non-serializable. A set of subgoals is independent if the set of operators can be partitioned into subsets such that the operators in a given subset are only relevant to a single subgoal; they are serializable if there exists an ordering such that they can be solved sequentially without ever violating previously solved subgoals; and they are non-serializable if they are not serializable.

There are some problems with Korf's taxonomy that make it difficult to use in planning. It is usually hard to know which class of the taxonomy a set of subgoals belongs to. More importantly, it lacks locality in the following sense. Suppose that the set $\{g_1, g_2\}$ of subgoals is serializable with the order of g_1 before g_2. Then to achieve the joint goal g_1 and g_2, a planner can achieve first g_1, then g_2. However, if the goal is to achieve jointly g_1, g_2, and g_3, then the planner can no longer use the strategy because the presence of the new subgoal g_3 may require the planner to achieve first g_2, then g_3, and finally g_1. For instance, this will be the case if g_2 is a precondition of any action that achieves g_3, while at the same time g_3 is a precondition of any action that achieves g_1.

In this paper, using the aforementioned isomorphism, we shall define in the situation calculus an ordering on goals that is independent of the context in which these goals appear, and apply it to planning. This paper is organized as follows.

In the next section, we briefly review the situation calculus, and make precise the foundational axioms that we will be using in this paper. Then, in section 3, we define some basic concepts such as goals and plans. We define in section 4 the ordering, and prove some basic properties about it. We then consider in section

5 algorithms for using the ordering in both linear and nonlinear planning. As it turns out, similar orderings have been proposed in the literature, in particular those of Cheng and Irani's [7, 2]. We discuss this and other related work in section 7. Finally, we make some concluding remarks in section 8.

2 Logical Preliminaries

2.1 The Language of the Situation Calculus

The language of the situation calculus is a many-sorted first-order one with the sorts *situation* for situations, *action* for actions, and *object* for everything else. It has the following domain independent predicates and functions: a constant S_0 of sort *situation* denoting the initial situation; a binary function $do(a, s)$ denoting the situation resulting from performing action a in situation s; a binary predicate $Poss(a, s)$ meaning that action a is possible (executable) in situation s; and a binary predicate $<$: $situation \times situation$. We shall follow convention, and write $<$ in infix form. By $s < s'$ we mean that s' can be obtained from s by a sequence of executable actions. As usual, $s \leq s'$ will be a shorthand for $s < s' \vee s = s'$.

Following convention, we define a *fluent* to be a predicate symbol of arity $object^n \times situation$ for some $n \geq 0$. For instance, the fluent $color : object^2 \times situation$ can be used to represent the color of a block: $color(x, y, s)$ holds if the block x is of the color y in the situation s.

In this paper, we shall assume the following foundational axioms Σ (Lin and Reiter [11]):[1]

$$S_0 \neq do(a, s),$$

$$do(a_1, s_1) = do(a_2, s_2) \supset (a_1 = a_2 \wedge s_1 = s_2),$$

$$(\forall P)[P(S_0) \wedge (\forall a, s)(P(s) \supset P(do(a, s))) \supset (\forall s)P(s)],$$

$$\neg s < S_0,$$

$$s < do(a, s') \equiv (Poss(a, s') \wedge s \leq s').$$

The first two axioms are unique names assumptions. They eliminate cycles, and avoid merging. The third axiom is second order induction. It amounts to the domain closure axiom that every situation has to be obtained from the initial one by repeatly applying the function do. As we shall see, induction will play an important role in this paper.[2] The last two axioms define $<$ inductively.

[1] These axioms have their origin in (Reiter [17]). Similar axioms are used in (Pinto and Reiter [16]).

[2] For a detailed discussion of the use of induction in the situation calculus, see (Reiter [17]).

Notice the similarity between these axioms and Peano foundational axioms for number theory. However, unlike Peano arithmetic which has a unique successor function, we have a class of successor functions here represented by the function do.

3 Simple Goals, Goals, and Plans

We now define some basic concepts of classical planning. We define a *simple goal* g to be an expression of the form $F(t_1, ..., t_n)$, where F is a fluent of arity $object^n \times situation$, and $t_1, \ldots, t_n$ are terms of sort $object$. Notice that a simple goal is *not* a formula in the situation calculus. It is an expression obtained from a fluent atom by suppressing its situation argument. We define a *goal* G to be an expression of the form $g_1 \ \& \ \cdots \ \& \ g_n$, where $g_1, \ldots, g_n$ are simple goals.

Let g be a simple goal, and st a situation term. We use $g[st]$ to denote the formula obtained from g by putting st back as its last argument. Therefore if g is $F(t_1, ..., t_n)$, then $g[st]$ is $F(t_1, ..., t_n, st)$. Notice that $g[st]$ is an atomic formula of our language. Now if the goal G is $g_1 \ \& \ \cdots \ \& \ g_n$, then we define $G[st]$ to be the formula $g_1[st] \wedge \cdots \wedge g_n[st]$.

Given a situation calculus theory $\mathcal{D}$ and a goal G, a finite sequence of actions $\Gamma = \alpha_1; \cdots; \alpha_n$ is called a *plan* for G iff

$$\mathcal{D} \models (\forall \vec{x}).S_0 \leq do(\Gamma, S_0) \wedge G[do(\Gamma, S_0)],$$

where $\vec{x}$ is the tuple of the free variables in G and Γ, and $do(\Gamma, S_0)$ is defined, recursively, as: $do(\emptyset, s) = s$, and $do(\alpha; \Gamma', s) = do(\Gamma', do(\alpha, s))$.

Notice that in the definition we need the condition $S_0 \leq do(\Gamma, S_0)$ because we want Γ to be executable in S_0. We shall call a situation s such that $S_0 \leq s$ a *legal* situation. So, because of our second-order induction axiom on situations, plans and legal situations are isomorphic in a precise sense: A situation s is legal iff there is a unique plan Γ such that $s = do(\Gamma, S_0)$. In the following, we shall say that *a legal situation s is a plan for the goal G* if the plan that corresponds to s is a plan for G. Then, according to our definition, we have that a legal situation s is a plan for G iff $G[s]$ holds.

Another important consequence of the isomorphism between legal situations and plans is that our partial order $\leq$ on situations becomes prefix relation on plans: For any legal situations s and s', $s \leq s'$ iff the plan that corresponds to s is a prefix of the plan that corresponds to s'. So the assertion that the plan Γ always protects the goal g can be represented as the following formula in the situation calculus:

$$(\forall s).S_1 \leq s \leq do(\Gamma, S_0) \supset g[s].$$

Similarly, the assertion that the legal situation s first achieves g_1, then achieves g_2 while protecting g_1 can be represented as the following formula:

$$g_1 \ \& \ g_2[s] \wedge (\exists s').S_0 \leq s' < s \wedge g_1[s'] \wedge \neg g_2[s'] \wedge (\forall s'')(s' \leq s'' \leq s \supset g_1[s''])$$

4 A Context-Independent Ordering on Goals

We are now ready to define our ordering on (simple) goals. Let g_1 and g_2 be two simple goals, and st a situation term. We say that g_1 has *precedence* over g_2 in situation st, written $Precedence(g_1, g_2, st)$, if the following condition holds: Whenever the situation st is a plan for $g_1 \,\&\, g_2$, it is the case that either $g_1 \,\&\, g_2$ is always true, or the plan first achieves g_1, then achieves g_2 while protecting g_1:

$$\begin{aligned} &Precedence(g_1, g_2, st) \stackrel{\triangle}{=} \\ &\quad S_0 \leq st \wedge g_1 \,\&\, g_2[st] \supset \{(\forall s').S_0 \leq s' \leq st \supset g_1 \,\&\, g_2[s'] \vee \\ &\qquad (\exists s').S_0 \leq s' < st \wedge g_1[s'] \wedge \neg g_2[s'] \wedge (\forall s'')(s' \leq s'' \leq st \supset g_1[s''])\}. \end{aligned}$$

Technically, the above expression means that $Precedence(g_1, g_2, st)$ is a shorthand for the formula in the right hand side of $\stackrel{\triangle}{=}$. We cannot treat $Precedence$ as a predicate because its arguments g_1 and g_2 are not legal terms of our language.

The precedence relation we have just defined is with respect to a particular plan. For planning purpose, however, we do not really care how a particular plan achieves a goal. What we are looking for are patterns and regularities of a class of plans, in the hope that such patterns and regularities, once known, will be useful to the planner. Considering the class of all plans for $g_1 \,\&\, g_2$, we thus define the macro $g_1 \prec g_2$ as follows:

$$g_1 \prec g_2 \stackrel{\triangle}{=} (\forall s).S_0 < s \wedge g_1 \,\&\, g_2[s] \supset Precedence(g_1, g_2, s).$$

We can prove:

Theorem 1 *Let g_1 and g_2 be two simple goals. $g_1 \prec g_2$ iff for any action a and any legal situation s, if $do(a, s)$ is a plan for $g_1 \,\&\, g_2$, then s is a plan for g_1:*

$$\Sigma \models g_1 \prec g_2 \equiv (\forall a, s).S_0 < do(a, s) \wedge g_1 \,\&\, g_2[do(a, s)] \supset g_1[s].$$

This is the main theorem that we'll use for proving assertions about our ordering relation $\prec$. For instance, consider the usual blocks world with the following actions:

- $move(x, y, z)$: Move the block x from block y onto the block z, provided both x and z are clear, and x is on y.
- $stack(x, y)$: Stack the block x on the block y, provided the block x is on the table and clear, and y is clear.
- $unstack(x, y)$: Unstack the block x from the block y, and put it on the table, provided x is clear and on y.

Suppose the goal is to build a tower with A on B, and B on C:

$$on(A, B) \,\&\, on(B, C).$$

We claim that the subgoal $on(B, C)$ has to be achieved before the subgoal $on(A, B)$:

$$on(B, C) \prec on(A, B).$$

To prove this, by the theorem, we need to prove that for any situation s, and any action a, we have:

$$S_0 < do(a, s) \wedge on(B, C) \,\&\, on(A, B)[do(a, s)] \supset on(B, C)[s].$$

To prove this, we show that

$$S_0 < do(a, s) \wedge \neg on(B, C, s) \wedge on(B, C, do(a, s)) \supset \neg on(A, B, do(a, s)).$$

Now suppose that $\neg on(B, C, s) \wedge on(B, C, do(a, s))$. Then the action a has to be either $stack(B, C)$ or $move(B, C, z)$ for some z. In either case, the block B must be clear after a is performed, so $on(A, B, do(a, s))$ must not hold.

Of course, the above is only informal argument. In the full version of the paper, we'll show by induction over situations that as long as the initial situation satisfies

$$(\forall x) clear(x, S_0) \equiv \neg(\exists y) on(y, x, S_0),$$

the subgoal $on(B, C)$ always has precedence over the subgoal $on(A, B)$. In particular, this is the case for Sussman's anomaly. This illustrates a difference between our ordering relation and Korf's taxonomy of subgoal interactions, for as Korf ([9]) notes, in Sussman's anomaly, the set of subgoals $\{on(A, B), on(B, C)\}$ is not serializable.

5 Using the Goal Ordering in Planning

We now show how knowledge about $\prec$ can be used effectively in planning. We consider two planners. One is a linear regression planner adapted from (Genesereth and Nilsson [5]). The other is the nonlinear planner SNLP of McAllester and Rosenblitt [13].

For ease of presentation, we shall consider only context-free actions. In the situation calculus, these actions are specified by a *context-free action theory* $\mathcal{D}$ (cf. Lin and Reiter [10, 12]) of the following form:

$$\mathcal{D} = \Sigma \cup \mathcal{D}_{ss} \cup \mathcal{D}_{ap} \cup \mathcal{D}_{una} \cup \mathcal{D}_{S_0},$$

where

- Σ is the set of the foundational axioms in Section 2.1.

- $\mathcal{D}_{ss}$ is a set of *context free successor state axioms*, one for each fluent F, of the form:

$$\begin{aligned} &Poss(a,s) \supset F(\vec{x}, do(a,s)) \equiv \\ &\quad (\exists \vec{u}) a = A_1(\vec{x},\vec{u}) \vee \cdots \vee (\exists \vec{v}) a = A_m(\vec{x},\vec{v}) \vee \\ &\quad F(\vec{x},s) \wedge \neg(\exists \vec{w}) a = B_1(\vec{x},\vec{w}) \wedge \cdots \wedge \neg(\exists \vec{t}) a = B_n(\vec{x},\vec{t}). \end{aligned} \tag{1}$$

  Here, the A's are the actions that have a *positive effect* on F, and the B's are the actions that have a *negative effect* on F. Other actions have *no effect* on F.

- $\mathcal{D}_{ap}$ is a set of *action precondition axioms*, one for each action A, of the form:

$$Poss(A(\vec{x}), s) \equiv F_1(\vec{t}_1, s) \wedge \cdots \wedge F_n(\vec{t}_n, s) \wedge E, \tag{2}$$

  where $F_1, \ldots, F_n$ are fluents, E is a propositional formula constructed from equality literals, and all free variables in $\vec{t}_1, \ldots, \vec{t}_n$ and E are in $\vec{x}$. In the following, we call E the *equality constraint* of the action A.

- $\mathcal{D}_{una}$ is the set of unique names axioms for actions:

$$\begin{aligned} &A(\vec{x}) \neq A'(\vec{y}), \quad \text{provided } A \text{ and } A' \text{ are distinct action names} \\ &A(x_1, ..., x_n) = A(y_1, ..., y_n) \supset x_1 = y_1 \wedge ... \wedge x_n = y_n. \end{aligned}$$

- $\mathcal{D}_{S_0}$, the initial database, is a finite set of first-order sentences that do not mention $<$, $Poss$, and any situation term except the constant S_0.

5.1 A Linear Regression Planner

We consider here a linear regression planner adapted from (Genesereth and Nilsson [5]). Given a goal G_0, the planner can be described as follows:

1. Initialize Γ (for partial plans) to $\emptyset$, and K (for subgoals) to the goal G_0.
2. If K is satisfied in the initial situation, then output Γ as the final plan.
3. Otherwise, choose a subgoal g in K, and choose an action A that has a positive effect on g. Append A to Γ, and regress K through the action A to get a new set of goals.
4. Go back to step 2.

It can be shown that the planner is sound and complete in the following sense: There is a plan for G_0 iff there is an execution of this procedure that output the plan as Γ.

Now if there is knowledge available about precedences among subgoals, then step 3 can be easily revised to take this knowledge into account without sacrificing the soundness and the completeness of the planner:

3′ Otherwise, choose a subgoal g in K *such that there is no other subgoal* g' *in* K *such that* $g \prec g'$ *holds*, and choose an action A that has a positive effect on g. Append A to Γ, and regress K through the action A to get a new set of goals.

The intuition here is that since the regression planner works backward from the goal, if $g_1 \prec g_2$, then the planner should work on g_2 first. For instance, consider again the blocks world, and the goal $on(A, B) \,\&\, on(B, C)$. Since $on(B, C) \prec on(A, B)$ holds, the planner will work on $on(A, B)$ first. Suppose it chooses the action $stack(A, B)$ to achieve this simple goal, then the original goal will be regressed to

$$on(B, C) \,\&\, clear(A) \,\&\, clear(B) \,\&\, ontable(A).$$

Since $clear(B) \prec on(B, C)$ is the only precedence relation that holds among the new subgoals, the planner can now choose to work on either $clear(B)$, or $clear(A)$, or $ontable(A)$, but not on $on(B, C)$.

5.2 A Nonlinear Planner

We shall now describe a nonlinear planner that can make use of knowledge about the precedence relation $\prec$ to minimize potential threats during planning.

Our starting point is the observation that if the subgoal g_1 has precedence over the subgoal g_2, then a smart planner should not treat the two subgoals as independent, and work on them separately. Instead, the planner should group them into a single goal, and consider only those actions that have a positive effect on g_2 by assuming that g_1 has already been achieved. Formally, we have the following result:

Theorem 2 *Let $\mathcal{D}$ be a context free action theory as described above. Let g_1 and g_2 be two ground simple goals such that $\mathcal{D} \models g_1 \prec g_2$. Let $A_1, \ldots, A_m$ be the actions that have a positive effect on g_2 but no effect on g_1, and $B_1, \ldots, B_n$ the actions that have a negative effect on either g_1 or g_2, then*

$$\begin{aligned}\mathcal{D} \models (\forall a, s).S_0 \leq s \wedge Poss(a, s) \supset \{&g_1 \,\&\, g_2[do(a, s)] \equiv \\ &g_1[s] \wedge [(\exists \vec{u})a = A_1(\vec{u}) \vee \cdots \vee (\exists \vec{v})a = A_m(\vec{v})] \vee \\ &g_1 \,\&\, g_2[s] \wedge \neg(\exists \vec{w})a = B_1(\vec{w}) \wedge \cdots \wedge \neg(\exists \vec{t})a = B_n(\vec{t})\}.\end{aligned}$$

This theorem can be extended to cases where there are more than two subgoals.

We now describe a nonlinear planner based on this theorem. It is designed after McAllester and Rosenblitt's SNLP ([13]).

We define a *nonlinear plan* with respect to a context free action theory $\mathcal{D}$ to be a tuple $(G_0, \mathcal{S}, \tau, \mathcal{O}, \mathcal{L})$, where

1. G_0 is a ground goal.

2. $\mathcal{S} \cup \{\texttt{START}, \texttt{FINISH}\}$ is a finite set whose elements are called *step names*, and $\mathcal{S} \cap \{\texttt{START}, \texttt{FINISH}\} = \emptyset$.
3. τ is a function that interprets $\mathcal{S}$ by assigning each of its elements a ground action term.
4. $\mathcal{O}$ is a set of *safety conditions* of the form $w < w'$, where w and w' are step names.
5. $\mathcal{L}$ is a set of causal links of the form (w, g, G, w'), where
   (a) w and w' are distinct step names; G is a goal, and g is a simple goal in G.
   (b) If $w = \texttt{START}$, then $\mathcal{D}_{S_0} \models G[S_0]$. Recall that $\mathcal{D}_{S_0}$ is the initial database in $\mathcal{D}$.
   (c) If $w \in \mathcal{S}$, then $\tau(w)$ has a positive effect on g, and has no effect on other simple goals in G.

Intuitively, a causal link (w, g, G, w') means that: (a) G is a set of prerequisites for the step w'; (b) the planner has decided to work on G as a whole; (c) the planner picks the subgoal g in G to work on; and (d) the planner decides to support g using the step w.

The main difference between our definition of a causal link and McAllester and Rosenblitt's is the extra G in our definition. We need the more general notion because our planner may decide to work on a set of simple goals in a coordinative way.

Given a nonlinear plan $\beta = (G_0, \mathcal{S}, \tau, \mathcal{O}, \mathcal{L})$ and a step w, a simple goal g is called a *prerequisite* of a step w if one of the following conditions holds:

1. $w = \texttt{FINISH}$, and g is in G_0.
2. $w \in \mathcal{S}$, and $g[s]$ is a conjunct of $Poss(\tau(w), s)$.
3. $w \neq \texttt{START}$, and there is a causal link (w, g', G, w') in $\mathcal{L}$ such that g is in G, but distinct from g'.

The last case is needed because the step w in (w, g', G, w') only establishes g', so the other subgoals in G have to be regressed to w.

Our nonlinear planner will output *complete nonlinear plans*, which are nonlinear plans $\beta = (G_0, \mathcal{S}, \tau, \mathcal{O}, \mathcal{L})$ such that:

1. For any step name w, if g is a prerequisite of w, then $\mathcal{L}$ contains a causal link of the form (w', g', G, w) such that g is in G. Notice that g' may be different from g.
2. If $v \in \mathcal{S}$ is a *threat* to $(w, g, G, w') \in \mathcal{L}$, then either $v < w \in Closure(\mathcal{O})$ or $w < v \in Closure(\mathcal{O})$.

3. For any step name $w \in \mathcal{S}$, *the equality constraint of the action* $\tau(A)$ *is satisfied*, i.e. if $Poss(\tau(w), s)$ is $g_1[s] \wedge \cdots \wedge g_n[s] \wedge E$, then $\mathcal{D}_{S_0} \models E$.

Here, $Closure(\mathcal{O})$ is basically the transitive closure of $\mathcal{O}$, and *threats* are defined as usual: a step v in $\mathcal{S}$ is called a *threat* to the causal links (w, g, G, w') if v is distinct from w and w', and $\tau(v)$ has an effect on g.

It can be shown that if a nonlinear plan is both complete and *order consistent* in the sense that there are no w such that $w < w$ is in $Closure(\mathcal{O})$, then any linearization of this nonlinear plan is a plan for G_0.

We can now describe our planner. Let β be $(G_0, \mathcal{S}, \tau, \mathcal{O}, \mathcal{L})$, and initialize $\mathcal{S}$, τ, $\mathcal{O}$, and $\mathcal{L}$ to $\emptyset$:

1. If β is order inconsistent, then fail.
2. If β is complete, then return β.
3. If there is a link (w, g, G, w'), and a threat v to this link in β such that neither $v < w$ nor $w' < v$ is in $Closure(\mathcal{O})$, then nondeterministically add either $v < w$ or $w' < v$ to $\mathcal{O}$. Go back to Step 1.
4. There must now exist some step name w in $\mathcal{S}$ such that at least one of its prerequisites are still not supported, i.e. the set

$$Open(w) = \{g \mid g \text{ is a prerequisite of } w \text{ and there is no link } (w', g', G, w) \text{ in } \mathcal{L} \text{ such that } g \text{ is in } G\}$$

is not empty. In this case, let

$$g_{1k} \prec \cdots \prec g_{12} \prec g_{11} \prec g, \cdots, g_{mn} \prec \cdots \prec g_{m2} \prec g_{m1} \prec g,$$

$k, m, n \geq 0$, be subgoals in $Open(w)$, and let G be

$$g \,\&\, g_{11} \,\&\, \cdots \,\&\, g_{1k} \,\&\, \cdots \,\&\, g_{m1} \,\&\, \cdots \,\&\, g_{mn}.$$

If at least one of the following steps succeed, then nondeterministically do one of them and then go back to Step 1; otherwise fail:

(a) Let w' be an existing step name such that $\tau(w')$ has a positive effect on g, but no effect on others in G. Add the link (w', g, G, w) to $\mathcal{L}$.

(b) If $\mathcal{D}_{S_0} \models G[S_0]$, then add the link $(\texttt{START}, g, G, w)$ to $\mathcal{L}$.

(c) Select an action α which has a positive effect on g, but no effect on others in G, and whose equality constraint is satisfied. Create a new step name w'. Add w' to $\mathcal{S}$. Let $\tau(w')$ be α. Add the link (w', g, G, w) to $\mathcal{L}$.

It can be shown that this planner inherits many properties of SNLP. For instance, it is also sound and complete for ground goals. Let us now illustrate this planner using a machine shop scheduling domain adapted from (Smith and Peot [18]). There are three actions the agent can perform:

- $shape(x)$: shape x and undo the effect of $drill(x)$, provided x is free.
- $drill(x)$: drill x, provided x is free.
- $bolt(x, y)$: fasten x to y, provided both x and y are drilled.

This domain can be axiomatized by a context-free action theory as follows:

Action precondition axioms $\mathcal{D}_{ap}$:

$$Poss(shape(x), s) \equiv free(x, s),$$
$$Poss(drill(x), s) \equiv free(x, s),$$
$$Poss(bolt(x, y), s) \equiv drilled(x, s) \wedge drilled(y, s) \wedge x \neq y.$$

Successor state axioms $\mathcal{D}_{ss}$:

$$Poss(a, s) \supset \{shaped(x, do(a, s)) \equiv a = shape(x) \vee shaped(x, s)\},$$

$$Poss(a, s) \supset \{drilled(x, do(a, s)) \equiv$$
$$a = drill(x) \vee [drilled(x, s) \wedge a \neq shape(x)]\},$$

$$Poss(a, s) \supset \{fastened(x, y, do(a, s)) \equiv$$
$$a = bolt(x, y) \vee a = bolt(y, x) \vee fastened(x, y, s)\},$$

$$Poss(a, s) \supset \{free(x, do(a, s)) \equiv$$
$$free(x, s) \wedge \neg(\exists y)a = bolt(x, y) \wedge \neg(\exists y)a = bolt(y, x)\}.$$

The initial database $\mathcal{D}_{S_0}$:

$$A \neq B,\ (\forall x)\neg shaped(x, S_0),\ (\forall x)\neg drilled(x, S_0),\ (\forall x, y)\neg fastened(x, y, S_0),$$
$$(\forall x).free(x, S_0) \equiv \neg(\exists y)fastened(x, y, S_0).$$

For this example, it is clear that to achieve $shaped(x)$ & $drilled(x)$, we have to achieve $shaped(x)$ first because the only action ($shape(x)$) that can make $shaped(x)$ true makes $drilled(x)$ false. So we have $shaped(x) \prec drilled(x)$. Similarly, we can show that if $x \neq y$, then:

$$drilled(x) \prec fastened(x, y), \quad drilled(y) \prec fastened(x, y),$$
$$shaped(x) \prec fastened(x, y), \quad shaped(y) \prec fastened(x, y).$$

Now given the goal $G_0 = shaped(A)$ & $shaped(B)$ & $fastened(A, B)$, our nonlinear planner works as follows. Since

$$shaped(A) \prec fastened(A, B) \wedge shaped(B) \prec fastened(A, B)$$

holds, so the planner chooses an action that has a positive effect on $fastened(A, B)$ but no effect on $shaped(A)$ or $shaped(B)$. The only action with this property is $bolt(A, B)$. So it creates a step w_1 for this action. This yields the causal link:

$$(w_1, fastened(A, B), shaped(A) \,\&\, shaped(B) \,\&\, fastened(A, B), \texttt{FINISH}).$$

Now the planner has to achieve the prerequisites of w_1, which are

$$shaped(A), shaped(B), drilled(A), drilled(B).$$

Since both $shaped(A) \prec drilled(A)$ and $shaped(B) \prec drilled(B)$ hold, it can work on either $shaped(A) \,\&\, drilled(A)$ or $shaped(B) \,\&\, drilled(B)$. Suppose it decides to work on the latter. Since $drill(B)$ is the only action that has a positive effect on $drilled(B)$ but no effect on $shaped(B)$, it creates a step w_2 for this action, and adds the following new causal link:

$$(w_2, drilled(B), shaped(B) \,\&\, drilled(B), w_1).$$

Now the planner has to achieve the prerequisites of w_2, which are $shaped(B)$ and $free(B)$. They can be achieved by adding the following two links:

$$(\texttt{START}, free(B), free(B), w_2),\ (w_3, shaped(A), shaped(A), w_2),$$

where the new step w_3 is mapped to $shape(A)$. Similarly, for the prerequisite $free(B)$ of w_3, it adds the link:

$$(\texttt{START}, free(B), free(B), w_3).$$

Similar causal links can be added to achieve the other pair of prerequisites, $shaped(A)$ and $drilled(A)$, of w_1. When this is done, it has a complete, order consistent nonlinear plan.

Notice that for this example, as long as the planner makes use of knowledge about $\prec$, no threat removal strategies are needed. For SNLP, as shown in (Smith and Peot [18]), some non-trivial threat removal strategies are needed in order to avoid backtracking.

6 Related Work

The situation calculus has been used for planning ever since it's introduced (McCarthy [14], McCarthy and Hayes [15], Green [6]). However, to the best of our knowledge, this paper is the first attempt in using the situation calculus to formalize control knowledge in planning.

We have pointed out some relationships between our ordering and Korf's taxonomy of subgoal interactions. Similar remarks can be made about the work of Joslin and Roach ([8]), and that of Barrett and Weld ([1]).

Although developed independently, it turns out that our ordering relation is closely related to that in (Etzioni [3]) and that in (Cheng and Irani [2]). It is fair to say that the intuitions behind all of these orderings are similar. The differences are mainly technical. For instance, although we consider only the space of *legal*

situations, Cheng and Irani ([2]) consider the space of all possible situations.[3] This difference is important. For instance, without restricting to legal situations, we would not be able to show that $on(B, C) \prec on(A, B)$ holds for the blocks world. For instance, given a situation s such that

$$clear(B, s) \wedge on(A, B, s) \wedge ontable(C, s) \wedge clear(C, s)$$

holds, the action $stack(C, B)$ will make both $on(A, B)$ and $on(B, C)$ true, thus achieves $on(B, C)$ while "protecting" $on(A, B)$.

7 Concluding Remarks

We have defined an ordering on goals in the situation calculus, and shown how knowledge about it can be used in a provably correct way in both linear and nonlinear planning.

Information on goal orderings is only one example of control knowledge that can be formalized in the situation calculus. The following are some more examples. We can define that the goal G_1 is *necessary* for achieving the goal G_2 iff

$$(\forall s).S_0 \leq s \wedge G_2[s] \supset (\exists s').S_0 < s' < s \wedge G_1[s'].$$

Similarly, we can say that the action A is *obligatory* for achieving the goal G iff

$$(\forall s).S_0 \leq s \wedge G[s] \supset (\exists s').S_0 < do(A, s') \leq s.$$

How to make use of this and other control information in planning is a research project that we are currently pursuing.

References

[1] A. Barrett and D. S. Weld. Characterizing subgoal interactions for planning. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence (IJCAI–93), IJCAI Inc. Distributed by Morgan Kaufmann, San Mateo, CA.*, pages 1388–1393, 1993.

[2] J. Cheng and K. B. Irani. Ordering problem subgoals. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI–89)*, pages 931–936, 1989.

[3] O. Etzioni. Acquiring search-control knowledge via static analysis. *Artificial Intelligence*, 62:255–301, 1993.

[3] It seems that Etzioni ([3]) has an implicit notion of legality built into his algorithms by using partial evaluation and state constraints.

[4] R. E. Fikes and N. J. Nilsson. STRIPS: A new approach to theorem proving in problem solving. *Artificial Intelligence*, 2:189–208, 1971.

[5] M. R. Genesereth and N. J. Nilsson. *The Logical Foundations of Artificial Intelligence*. Morgan Kaufmann, Los Altos, CA., 1987.

[6] C. C. Green. Application of theorem proving to problem solving. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI–69)*, pages 219–239, 1969.

[7] K. B. Irani and J. Cheng. Subgoal ordering and goal augmentation for heuristic problem solving. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence (IJCAI–87)*, pages 1018–1024, 1987.

[8] D. Joslin and J. Roach. A theoretical analysis of conjunctive-goal problems. *Artificial Intelligence*, 41:97–106, 1989.

[9] R. E. Korf. Planning as search: A quantitative approach. *Artificial Intelligence*, 33:65–88, 1987.

[10] F. Lin and R. Reiter. How to progress a database (and why) I. Logical foundations. In *Proceedings of the Fourth International Conference on Principles of Knowledge Representation and Reasoning (KR'94)*, pages 425–436, 1994.

[11] F. Lin and R. Reiter. State constraints revisited. *Journal of Logic and Computation, Special Issue on Actions and Processes*, 4(5):655–678, 1994.

[12] F. Lin and R. Reiter. How to progress a database II: The STRIPS connection. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI–95), IJCAI Inc. Distributed by Morgan Kaufmann, San Mateo, CA.*, pages 2001–2007, 1995.

[13] D. McAllester and D. Rosenblitt. Systematic nonlinear planning. In *Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI–91)*, pages 634–639, 1991.

[14] J. McCarthy. Situations, actions and causal laws. In M. Minsky, editor, *Semantic Information Processing*, pages 410–417. MIT Press, Cambridge, Mass., 1968.

[15] J. McCarthy and P. Hayes. Some philosophical problems from the standpoint of artificial intelligence. In B. Meltzer and D. Michie, editors, *Machine Intelligence 4*, pages 463–502. Edinburgh University Press, Edinburgh, 1969.

[16] J. Pinto and R. Reiter. Extending the situation calculus with event occurrences. In *Second Symposium on Logical Formalizations of Commonsense Reasoning*, 1993.

[17] R. Reiter. Proving properties of states in the situation calculus. *Artificial Intelligence*, 64:337–351, 1993.

[18] D. E. Smith and M. A. Peot. Postponing threats in partial-order planning. In *Proceedings of the Eleventh National Conference on Artificial Intelligence (AAAI–93), AAAI Press, Menlo Park, CA.*, pages 500–506, 1993.