An Ordering on Subgoals for Planning

Fangzhen Lin
Department of Computer Science
The Hong Kong University of Science and Technology
Clear Water Bay, Kowloon, Hong Kong
fl@cs.ust.hk

Abstract

Subgoal ordering is a type of control information that has received much attention in AI planning community. In this paper we formulate precisely a subgoal ordering in the situation calculus. We show how information about this subgoal ordering can be deduced from the background action theory. We also show for both linear and nonlinear planners how knowledge about this ordering can be used in a provably correct way to avoid unnecessary backtracking.

1 Introduction

In a typical AI planning problem (cf. Chapman [3]), a goal is represented as a conjunction of simpler subgoals. The problem of subgoal ordering is about in which order a planner should attempt to achieve these subgoals. For example, given the goal of having block A on top of B and block B on top of C, any plan that achieves this goal will have to achieve the subgoal of having B on C first, assuming of course the usual constraints on the blocks world.

The problem of subgoal ordering is important for AI planning. Korf [9] first shows that the complexity of a planning problem is closely related to the question of whether there is an effective ordering on subgoals (see also Barrett and Weld [2] and Joslin and Roach [8]).

In this paper we formulate a subgoal ordering in the situation calculus. We show how information about this subgoal ordering can be deduced from the background action theory. We also show for both linear and nonlinear planners how knowledge about this ordering can be used in a provably correct way to avoid unnecessary backtracking.

The rest of this paper is organized as follows. We first provide necessary background about the situation calculus in section 2. Then in section 3 we define the basic concepts in planning that will be used in this paper. In section 4 we precisely define in the situation calculus our subgoal ordering. In section 5 we show how to compute this ordering given

a background action theory. In section 6 we show how knowledge about this ordering can be used in a provably correct way in planning. In section 7 we make some general remarks about our subgoal ordering and speculate on possible ways this ordering can be generalized. In section 9 we discuss some related work. Finally in section 9 we conclude this paper and point out some other control information that can be effectively formalized in the situation calculus.

2 The Situation Calculus

The situation calculus (McCarthy and Hayes [15]) is a formalism for representing and reasoning about actions in dynamic domains. It is a many-sorted predicate calculus with some reserved predicate and function symbols. For example, to say that block A is initially clear, we write:

$$H(clear(A), S_0),$$

where H is a reserved binary predicate and stands for "holds", and S_0 is a reserved constant symbol denoting the initial situation. As an another example, to say that the action $stack(x, y)$ causes $on(x, y)$ to be true, we write:[1]

$$Poss(stack(x, y), s) \supset H(on(x, y), do(stack(x, y), s)),$$

where the reserved function $do(a, s)$ denotes the resulting situation of doing the action a in the situation s, and the reserved predicate $Poss(a, s)$ is the precondition for a to be executable in s. This is an example of how the effects of an action can be represented in the situation calculus. Generally, in the situation calculus:

- situations are first-order objects that can be quantified over;
- a situation carries information about its history, i.e the sequence of actions that have been performed so far. For example, the history of the situation

  $$do(stack(A, B), do(stack(B, C), S_0))$$

  is $[stack(B, C), stack(A, B)]$, i.e. the sequence of actions that have been performed in the initial situation to reach this situation. As we shall see later, our foundational axioms shall enforce a one-to-one correspondence between situations and sequences of actions.

We believe that these two features of the situation calculus make it a natural formalism for representing and reasoning about control knowledge. For example, in AI planning, a plan is a sequence of actions, thus isomorphic to situations. So control knowledge in planning, which often are constraints on desirable plans, becomes constraints on situations.

[1] In this paper, free variables in a displayed formula are assumed to be universally quantified.

Formally, the language of the situation calculus is a many-sorted first-order one with equality. We assume the following sorts: *situation* for situations, *action* for actions, *fluent* for propositional fluents such as *clear* whose truth values depend on situations, and *object* for everything else. As we mentioned above, we assume that S_0 is a reserved constant denoting the initial situation, H a reserved predicate for expressing properties about fluents in a situation, *do* a reserved binary function denoting the result of performing an action, and *Poss* a reserved binary predicate for action preconditions. In addition, we assume a partial order $<$ on situations. Following convention, we 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 $s < s' \vee s = s'$.

In this paper, we assume the following foundational axioms Σ (Lin and Reiter [11]):[2]

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

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

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

$$\neg s < S_0,$$

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

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.[3] The last two axioms define $<$ inductively.

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*. The following proposition summarizes some simple consequences of Σ. Proofs can be found in (Lin and Reiter [11]).

Proposition 2.1

Transitivity: $(s_1 < s_2 \wedge s_2 < s_3) \supset s_1 < s_3$.

Anti-reflexivity: $\neg s < s$.

Unique names: $s_1 < s_2 \supset s_1 \neq s_2$.

Induction on $<$: $(\forall P)[P(S_0) \wedge (\forall a, s)(P(s) \wedge Poss(a, s) \supset P(do(a, s))) \supset (\forall s)(S_0 \leq s \supset P(s))]$.

[2] These axioms have their origin in (Reiter [20]). Similar axioms are used in (Pinto and Reiter [18]).

[3] For a detailed discussion of the use of induction in the situation calculus, see (Reiter [20]).

3 Simple Goals, Goals, and Plans

We now define some basic concepts of classical planning. We define a *simple goal (subgoal)* g to be a fluent term $F(t_1, ..., t_n)$, where F is a fluent of arity n, and $t_1, \ldots, t_n$ are terms of sort *object*. 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 $G = g_1 \,\&\, \cdots \,\&\, g_n$ be a goal, and S a situation term. We define $H(G, S)$, the truth value of G in S, to be the situation calculus formula: $H(g_1, S) \wedge \cdots \wedge H(g_n, S)$.

Given a situation calculus theory $\mathcal{D}$ and a goal G, a finite sequence of actions Γ is called a *plan* for G iff

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

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

An 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:

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

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:

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

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 S a situation term. We say that g_1 has *precedence* over g_2 in situation S, written $Precedence(g_1, g_2, S)$, if the following condition holds: Whenever the situation S 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:

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

Technically, the above expression means that $Precedence(g_1, g_2, S)$ is a shorthand for the formula in the right hand side of $\stackrel{\Delta}{=}$.

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 make the search for a plan more focus. Considering the class of all plans for $g_1 \,\&\, g_2$, we thus define the macro $g_1 \prec g_2$ as follows:

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

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:*

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

Proof: By our definition, $g_1 \prec g_2$ is

$$(\forall s).S_0 < s \wedge H(g_1 \,\&\, g_2, s) \supset Precedence(g_1, g_2, s).$$

By expanding $Precedence$, this is equivalent to

$$\begin{aligned}(\forall s).S_0 < s \wedge H(g_1 \,\&\, g_2, s) \supset \{&(\forall s')[S_0 \leq s' \leq s \supset H(g_1 \,\&\, g_2, s')] \vee \\ &(\exists s').S_0 \leq s' < s \wedge H(g_1, s') \wedge \neg H(g_2, s') \wedge (\forall s'').s' \leq s'' \leq s \supset H(g_1, s'')\}.\end{aligned}$$

By induction on situations, we can show that for any formula $P(s)$, whenever there is a situation s such that $P(s)$ holds, then there must be a least such s. From this fact we can show that

$$(\exists s').S_0 \leq s' < s \wedge H(g_1, s') \wedge \neg H(g_2, s') \wedge (\forall s'').s' \leq s'' \leq s \supset H(g_1, s'')$$

is equivalent to

$$\begin{aligned}(\exists a, s').S_0 \leq do(a, s') \leq s \wedge H(g_1, s') \wedge \neg H(g_2, s') \wedge H(g_1 \,\&\, g_2, do(a, s')) \wedge \\ (\forall s'').do(a, s') \leq s'' \leq s \supset H(g_1 \,\&\, g_2, s'').\end{aligned}$$

Thus $g_1 \prec g_2$ is equivalent to

$$\begin{aligned}(\forall s).S_0 < s \wedge H(g_1 \,\&\, g_2, s) \supset \{&(\forall s')[S_0 \leq s' \leq s \supset H(g_1 \,\&\, g_2, s')] \vee \\ &(\exists a, s').S_0 \leq do(a, s') \leq s \wedge H(g_1, s') \wedge \neg H(g_2, s') \wedge H(g_1 \,\&\, g_2, do(a, s')) \wedge \\ &(\forall s'').do(a, s') \leq s'' \leq s \supset H(g_1 \,\&\, g_2, s'')\}. \qquad (1)\end{aligned}$$

From this it is easy to see that

$$\Sigma \models g_1 \prec g_2 \supset \{(\forall a, s).S_0 < do(a, s) \wedge H(g_1 \,\&\, g_2, do(a, s)) \supset H(g_1, s)\}.$$

To prove the other half of the equivalence, suppose that

$$(\forall a, s).S_0 < do(a, s) \wedge H(g_1 \,\&\, g_2, do(a, s)) \supset H(g_1, s) \tag{2}$$

holds. To prove (1), assume that there is a situation s' such that

$$S_0 < s \wedge H(g_1 \,\&\, g_2, s) \wedge S_0 \leq s' < s \wedge \neg H(g_1 \,\&\, g_2, s').$$

By induction, there is a situation s'' and an action a such that

$$s' \leq s'' < do(a, s'') \leq s \wedge \neg H(g_1 \,\&\, g_2, s'') \wedge (\forall s^*).do(a, s'') \leq s^* \leq s \supset H(g_1 \,\&\, g_2, s^*).$$

By (2), this implies that

$$s' \leq s'' < do(a, s'') \leq s \wedge H(g_1, s'') \wedge \neg H(g_2, s'') \wedge (\forall s^*).do(a, s'') \leq s^* \leq s \supset H(g_1 \,\&\, g_2, s^*).$$

Thus (2) implies that

$$\begin{array}{c} S_0 < s \wedge H(g_1 \,\&\, g_2, s) \wedge (\exists s')[S_0 \leq s' \leq s \wedge \neg H(g_1 \,\&\, g_2, s')] \supset \\ (\exists a, s'').S_0 \leq do(a, s'') \leq s \wedge H(g_1, s'') \wedge \neg H(g_2, s'') \wedge \\ (\forall s^*).do(a, s'') \leq s^* \leq s \supset H(g_1 \,\&\, g_2, s^*). \end{array}$$

So (2) implies (1). ■

This is our basic theorem about the relation $\prec$. In the following, we show how to use it to compute $\prec$ for a special class of action theories.

5 Computing the Subgoal Ordering in Action Theories with Successor State Axioms

A *basic action theory* $\mathcal{D}$ is one that has the following form (cf. Reiter [21] and Lin and Reiter [11]):

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

where

- Σ is the set of the foundational axioms for situations as given in Section 2.
- $\mathcal{D}_{ss}$ is a set of successor state axioms, one for each fluent F, of the form:

  $$Poss(a, s) \supset H(F(\vec{x}), do(a, s)) \equiv \Phi_F(\vec{x}, a, s),$$

  where $\Phi_F(\vec{x}, a, s)$ is a formula that does not mention $<$, $Poss$, and any situation term except possibly a situation variable s, and whose free variables are among $\vec{x}, a, s$. Intuitively, a successor state axiom for F defines the truth value of F in a successor situation in terms of the truth values of the fluents in the current situation.

- $\mathcal{D}_{ap}$ is a set of action precondition axioms of the form:

$$Poss(A(\vec{x}), s) \equiv \Psi_A(\vec{x}, s),$$

where A is an action, and $\Psi_A(\vec{x}, s)$ is a formula that does not mention $<$, $Poss$, and any situation term except possibly a situation variable s, and whose free variables are among $\vec{x}, s$.

- $\mathcal{D}_{una}$ is the set of unique names axioms for actions: For any two different actions $A(\vec{x})$ and $A'(\vec{y})$, we have

$$A(\vec{x}) \neq A'(\vec{y}),$$

and for any action $A(x_1, ..., x_n)$, we have

$$A(x_1, ..., x_n) = A(y_1, ..., y_n) \supset x_1 = y_1 \wedge ... \wedge x_n = y_n.$$

- $\mathcal{D}_{S_0}$, 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.

Example 5.1 Consider the following blocks world (Lin and Reiter [12]):

Actions[4]

- $move(x, y, z)$: Move block x from block y onto block z, provided both x and z are clear and block x is on top of block y.
- $movefromtable(x, y)$: Move block x from the table onto block y, provided x is clear and on the table, and block y is clear.
- $movetotable(x, y)$: Move block x from block y onto the table, provided x is clear and x is on top of y.

Fluents

- $clear(x)$: Block x is clear, i.e. has no other blocks on top of it.
- $on(x, y)$: Block x is on block y.
- $ontable(x)$: Block x is on the table.

This setting can be axiomatized as follows:

Action precondition axioms, one for each action:[5]

$$Poss(move(x, y, z), s) \equiv H(clear(x), s) \wedge H(clear(z), s) \wedge H(on(x, y), s) \wedge x \neq y \neq z,$$

[4]These actions are designed to make them context free. A more compact formalization will be to introduce $table$ as a constant, and replace $movetotable(x, y)$ and $movefromtable(x, y)$ by $move(x, y, table)$ and $move(x, table, y)$, respectively.

[5]In the following, the expression $x \neq y \neq z$ stands for $x \neq y \wedge x \neq z \wedge y \neq z$.

$$Poss(movefromtable(x,y),s) \equiv H(clear(x),s) \wedge H(clear(y),s) \wedge H(ontable(x),s) \wedge x \neq y,$$

$$Poss(movetotable(x,y),s) \equiv H(clear(x),s) \wedge H(on(x,y),s).$$

Successor state axioms, one for each fluent:

$$\begin{array}{l} Poss(a,s) \supset \\ \quad H(ontable(x),do(a,s)) \equiv (\exists y)a = movetotable(x,y) \vee \\ \quad\quad H(ontable(x),s) \wedge \neg(\exists y)a = movefromtable(x,y). \end{array}$$

$$\begin{array}{l} Poss(a,s) \supset \\ \quad H(on(x,y),do(a,s)) \equiv (\exists z)a = move(x,z,y) \vee a = movefromtable(x,y) \vee \\ \quad\quad H(on(x,y),s) \wedge a \neq movetotable(x,y) \wedge \neg(\exists z)a = move(x,y,z), \end{array}$$

$$\begin{array}{l} Poss(a,s) \supset \\ \quad H(clear(x),do(a,s)) \equiv (\exists y,z)a = move(y,x,z) \vee (\exists y)a = movetotable(y,x) \vee \\ \quad\quad H(clear(x),s) \wedge \neg(\exists y,z)a = move(y,z,x) \wedge \neg(\exists y)a = movefromtable(y,x), \end{array}$$

The initial database $\mathcal{D}_{S_0}$ can be any finite set of sentences about S_0. In the following, we shall take it to be the set of following axioms:

$$A \neq B \neq C,$$

$$H(ontable(x),S_0) \equiv x = A,$$

$$H(on(x,y),S_0) \equiv (x = B \wedge y = C) \vee (x = C \wedge y = A),$$

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

This initial database captures the initial situation in Sussman's anomaly. ■

To formulate our theorem for computing $g_1 \prec g_2$ in basic action theories, we need to introduce the notion of *regression* (cf. Waldinger [23], Pednault [17], and Reiter [19]) and *state constraints.*

Definition 5.1 *Let $\mathcal{D}$ be a basic action theory, and $\Psi(s)$ a formula that does not mention any other situation term except s. Let $\Psi(do(\alpha,s))$ be $\Psi(s)$ with s replaced by $do(\alpha,s)$. We define the* regression *of $\Psi(do(\alpha,s))$ under the set of successor state axioms in $\mathcal{D}$, written $\mathcal{R}[\Psi(do(\alpha,s))]$, to be the result of substituting $\Phi_F(t_1,...,t_n,\alpha,S)$ for every subformula of the form $H(F(t_1,...,t_n),do(\alpha,S))$ mentioned in $\Psi(do(\alpha,s))$, for every fluent F. Here, Φ_F is as in the successor state axiom for F in $\mathcal{D}$.*

If α is executable in s, then $\mathcal{R}[\Psi(do(\alpha,s))]$ is the weakest precondition for Ψ to be true after α:

Lemma 5.1 $\mathcal{D} \models (\forall a,s).Poss(a,s) \supset \Psi(do(a,s)) \equiv \mathcal{R}[\Psi(do(a,s))]$.

Example 5.2 Given the blocks world in Example 5.1, the regression of

$$H(on(x,y), do(move(x,z,y),s)) \wedge H(clear(x'), do(move(x,z,y),s))$$

is

$$\begin{array}{l}\{(\exists z')move(x,z,y) = move(x,z',y) \vee \\ \ move(x,z,y) = movefromtable(x,y) \vee \\ \ H(on(x,y),s) \wedge move(x,z,y) \neq movetotable(x,y) \wedge \\ \qquad \neg(\exists z')move(x,z,y) = move(x,y,z')\} \wedge \\ \{(\exists y',z')move(x,z,y) = move(y',x',z') \vee \\ \ (\exists y')move(x,z,y) = movetotable(y',x') \vee \\ \ H(clear(x'),s) \wedge \neg(\exists y',z')move(x,z,y) = move(y',z',x') \wedge \\ \qquad \neg(\exists y')move(x,z,y) = movefromtable(y',x')\}.\end{array}$$

Under $\mathcal{D}_{una}$, this is equivalent to

$$x' = z \vee [H(clear(x'),s) \wedge x' \neq y].$$

Therefore, if $move(x,z,y)$ is executable, then $on(x,y)$ & $clear(x')$ holds afterward iff either $z = x'$ or $clear(x')$ holds initially and $y \neq x'$. ■

Definition 5.2 *Given an action theory $\mathcal{D}$, a sentence of the form $(\forall s).s \geq S_0 \supset C(s)$ is a* state constraint *if it is entailed by $\mathcal{D}$.*

To prove that $(\forall s).s \geq S_0 \supset C(s)$ is a state constraint, we normally need to appeal to the principle of induction on $<$ in Proposition 2.1 (Reiter [20]).

Example 5.3 Continuing our discussion of the blocks world Example 5.1. Consider the sentence $(\forall s).s \geq S_0 \supset C(s)$, where $C(s)$ is

$$(\forall x).H(clear(x),s) \equiv \neg(\exists y)H(on(y,x),s).$$

We claim that $(\forall s).s \geq S_0 \supset C(s)$ is a state constraint. To prove this, we need to use induction. However, to our surprise, applying directly the principle of induction on $<$ in Proposition 2.1 does not work. It is well known that to prove an assertion by induction, sometimes it is necessary to apply induction on a more general assertion. In this case, we need to apply the principle of induction on $<$ in Proposition 2.1 to

$$(\forall s).s \geq S_0 \supset C(s) \wedge C'(s),$$

where $C'(s)$ is

$$(\forall x,y,z).H(on(y,x),s) \wedge H(on(z,x),s) \supset y = z.$$

We were surprised because this means that the successor state axiom for *clear* as given in Example 5.1 cannot be derived from the successor state axiom for *on* and the definition $(\forall s)(\forall x).clear(x,s) \equiv \neg(\exists y)on(y,x,s)$. To do that, we also need to assume the state constraint $(\forall s).s \geq S_0 \supset C'(s)$. ■

The following theorem will be our basis for establishing conditional precedences. In this theorem, the role of state constraint is to pre-compute induction by putting constraints on legal situations. The unique names axioms $\mathcal{D}_{una}$ are needed to simplify the result of regression, as we have seen above.

Theorem 2 *Let $\mathcal{D}$ be an action theory, $(\forall s).s \geq S_0 \supset C(s)$ a state constraint, and P a formula. If*

$$\mathcal{D}_{una} \cup \mathcal{D}_{ap} \models (\forall \vec{x}).P \supset \{(\forall a, s).C(s) \wedge Poss(a, s) \wedge \mathcal{R}[H(g_1 \,\&\, g_2, do(a, s))] \supset H(g_1, s)\},$$

then $\mathcal{D} \models (\forall \vec{x}).P \supset g_1 \prec g_2$, where $\vec{x}$ is the tuple of the free variables in P, g_1, and g_2.

Proof: Let

$$\mathcal{D}_{una} \cup \mathcal{D}_{ap} \models (\forall \vec{x}).P \supset \{(\forall a, s).C(s) \wedge Poss(a, s) \wedge \mathcal{R}[H(g_1 \,\&\, g_2, do(a, s))] \supset H(g_1, s)\}.$$

Then

$$\begin{aligned}&\mathcal{D}_{una} \cup \mathcal{D}_{ap} \cup \{(\forall s).s \geq S_0 \supset C(s)\} \models \\ &\quad (\forall \vec{x}).P \supset (\forall a, s).s \geq S_0 \wedge Poss(a, s) \wedge \mathcal{R}[H(g_1 \,\&\, g_2, do(a, s))] \supset H(g_1, s).\end{aligned}$$

Thus

$$\mathcal{D} \models (\forall \vec{x}).P \supset (\forall a, s).s \geq S_0 \wedge Poss(a, s) \wedge \mathcal{R}[H(g_1 \,\&\, g_2, do(a, s))] \supset H(g_1, s).$$

So by Lemma 5.1, and the axioms for $<$ in Σ,

$$\mathcal{D} \models (\forall \vec{x}).P \supset (\forall a, s).s < do(a, s) \wedge H(g_1 \,\&\, g_2, do(a, s)) \supset H(g_1, s).$$

Therefore by Theorem 1, $\mathcal{D} \models (\forall \vec{x}).P \supset g_1 \prec g_2$. ■

Notice that the theorem generalizes to cases with more than one state constraints because for any two state constraints $(\forall s).s \geq S_0 \supset C_1(s)$ and $(\forall s).s \geq S_0 \supset C_2(s)$, the sentence $(\forall s).s \geq S_0 \supset C_1(s) \wedge C_2(s)$ is also a state constraint.

Example 5.4 Continuing with our blocks world, we show that

$$\mathcal{D} \models (\forall x, y, z).x \neq y \neq z \supset on(y, z) \prec on(x, y). \tag{3}$$

We shall use the state constraint that we proved in Example 5.3: $(\forall s).s \geq S_0 \supset C(s)$, where $C(s)$ is

$$s \geq S_0 \supset (\forall x).H(clear(x), s) \equiv \neg(\exists y)H(on(y, x), s),$$

Let P in Theorem 2 be $x \neq y \neq z$. By the theorem, we need to show that

$$\begin{aligned}&\mathcal{D}_{una} \cup \mathcal{D}_{ap} \cup \{x \neq y \neq z\} \models \\ &C(s) \wedge Poss(a, s) \wedge \mathcal{R}[H(on(y, z), do(a, s)) \wedge H(on(x, y), do(a, s))] \supset H(on(y, z), s)\end{aligned} \tag{4}$$

Now $\mathcal{R}[H(on(y,z),do(a,s)) \wedge H(on(x,y),do(a,s))]$ is

$$\begin{aligned}
&(\exists z')a = move(y,z',z) \vee a = movefromtable(y,z) \vee \\
&\qquad H(on(y,z),s) \wedge a \neq movetotable(y,z) \wedge \neg(\exists z')a = move(y,z,z') \wedge \\
&(\exists z')a = move(x,z',y) \vee a = movefromtable(x,y) \vee \\
&\qquad H(on(x,y),s) \wedge a \neq movetotable(x,y) \wedge \neg(\exists z')a = move(x,y,z').
\end{aligned}$$

By $\mathcal{D}_{una}$ and $x \neq y \neq z$, this formula can be simplified to:

$$\begin{aligned}
&(\exists z')a = move(y,z',z) \wedge H(on(x,y),s) \vee \\
&a = movefromtable(y,z) \wedge H(on(x,y),s) \vee \\
&(\exists z')a = move(x,y,z') \wedge H(on(y,z),s) \vee \\
&a = movefromtable(x,y) \wedge H(on(y,z),s) \vee \\
&H(on(x,y),s) \wedge H(on(y,z),s) \wedge a \neq movetotable(y,z) \wedge \neg(\exists z')a = move(y,z,z') \wedge \\
&\qquad a \neq movetotable(x,y) \wedge \neg(\exists z')a = move(x,y,z').
\end{aligned}$$

The first two disjuncts correspond to the cases where the action a makes $on(y,z)$ true but has no effect on $on(x,y)$; the third and fourth disjuncts correspond to the cases where the action a makes $on(x,y)$ true; the last disjunct corresponds to the case where the action a has no effect on either $on(y,z)$ or $on(x,y)$. Therefore, to prove (4), the only non-trivial cases to consider are the third and fourth disjuncts:

$$\begin{aligned}
&\mathcal{D}_{una} \cup \mathcal{D}_{ap} \cup \{x \neq y \neq z, C(s), Poss(a,s)\} \models \\
&\qquad (\exists z')a = move(y,z',z) \wedge H(on(x,y),s) \supset H(on(y,z),s),
\end{aligned}$$

and

$$\begin{aligned}
&\mathcal{D}_{una} \cup \mathcal{D}_{ap} \cup \{x \neq y \neq z, C(s), Poss(a,s)\} \models \\
&\qquad a = movefromtable(y,z) \wedge H(on(x,y),s) \supset H(on(y,z),s).
\end{aligned}$$

To prove the first entailment, suppose that $a = move(y,z^*,z)$ and $H(on(x,y),s)$ holds. By the action precondition axiom for $move$, we have $H(clear(y),s)$. Thus by $C(s)$, we have $\neg(\exists y')H(on(y',y),s)$. This contradicts with the assumption that $H(on(x,y),s)$. This proves

$$\begin{aligned}
&\mathcal{D}_{una} \cup \mathcal{D}_{ap} \cup \{x \neq y \neq z, C(s), Poss(a,s)\} \models \\
&\qquad (\exists z')a = move(y,z',z) \wedge H(on(x,y),s),
\end{aligned}$$

thus the first entailment. The proof of the second entailment is similar. Therefore we have proved (4). So we have that

$$\mathcal{D} \models (\forall x,y,z).x \neq y \neq z \supset on(y,z) \prec on(x,y).$$

In particular, we have that

$$\mathcal{D} \models on(B,C) \prec on(A,B).$$

We remark that the general assertion (3) does not depend on the particular configuration of the initial situation. In fact, it holds as long as the initial database $\mathcal{D}_{S_0}$ contains the following two facts:

$$(\forall x,y,z).H(on(y,x),S_0) \wedge H(on(z,x),S_0) \supset y = z,$$

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

■

This example also illustrates a difference between our ordering relation and Korf's taxonomy of subgoal interactions. As Korf [9] noted, in Sussman's anomaly, the set $\{on(A,B), on(B,C)\}$ of subgoals is not serializable according to his definition. But $on(B,C)$ has precedence over $on(A,B)$ according to our definition.

6 Using the Subgoal 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 [6]). 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 [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.
- $\mathcal{D}_{ss}$ is a set of *context free successor state axioms*, one for each fluent F, of the form:

$$\begin{aligned} &Poss(a,s) \supset H(F(\vec{x}),do(a,s)) \equiv \\ &\quad (\exists \vec{u}_1)a = A_1(\vec{x},\vec{u}_1) \vee \cdots \vee (\exists \vec{u}_m)a = A_m(\vec{x},\vec{u}_m) \vee \\ &\quad H(F(\vec{x}),s) \wedge \neg(\exists \vec{w}_1)a = B_1(\vec{x},\vec{w}_1) \wedge \cdots \wedge \neg(\exists \vec{w}_n)a = B_n(\vec{x},\vec{w}_n). \end{aligned} \tag{5}$$

  Intuitively, the A's are those actions that make F true, and the B's are those that make F false.

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

$$Poss(A(\vec{x}),s) \equiv H(F_1(\vec{t}_1),s) \wedge \cdots \wedge H(F_n(\vec{t}_n),s) \wedge E, \tag{6}$$

  where $F_1,\ldots,F_n$ are fluents, and E, called an *equality constraint*, is a propositional formula constructed from equality literals. We require here that all free variables in $\vec{t}_1,\ldots,\vec{t}_n$ and E be in $\vec{x}$.

- $\mathcal{D}_{una}$ is the set of unique names axioms for actions as given in section 5.
- $\mathcal{D}_{S_0}$, 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.

In the following, given an action α, we say that it has a *positive effect* on $F(\vec{t})$ if for some $1 \leq i \leq m$, $\mathcal{D}_{una} \models (\exists \vec{u_i})\alpha = A_i(\vec{t}, \vec{u_i})$, where A_i is as in the successor state axiom (5) for F. Similarly, we say that α has a *negative effect* on the fluent atom $F(\vec{t})$ if for some $1 \leq i \leq n$, $\mathcal{D}_{una} \models (\exists \vec{w_i})\alpha = B_i(\vec{t}, \vec{w_i})$, where B_i is as in the successor state axiom (5) for F. We say an action has *no effect* on $F(\vec{t})$ if it has neither a positive nor a negative effect on it.

6.1 A Linear Regression Planner

Let $\mathcal{D}$ be a context free action theory as described above. Let G_0 be a goal. A naive version of our linear regression planner adapted from (Nilsson [16] and Genesereth and Nilsson [6]) can be described as follows:

1. Initialize Γ (for partial plans) to $\emptyset$, $\mathcal{G}$ (for outstanding goals) to G_0, and $\mathcal{E}$ (for equality constraints) to $true$.
2. If there is a variable substitution σ such that $\mathcal{D}_{S_0} \cup \mathcal{D}_{una} \models (H(\mathcal{G}, S_0) \wedge \mathcal{E}) \cdot \sigma$,[6] then output $\Gamma \cdot \sigma$ as the final plan.
3. Choose an action $A(\vec{t})$. Suppose the axiom for $Poss(A(\vec{t}), s)$ is as (6):

   $$H(F_1(\vec{t_1}), s) \wedge \cdots \wedge H(F_n(\vec{t_n}), s) \wedge E,$$

   and suppose that under the unique names axioms $\mathcal{D}_{una}$, $\mathcal{R}[H(\mathcal{G}, do(A(\vec{t}), s))]$ is equivalent to:[7]

   $$(E_1 \wedge H(G_1, s)) \vee \cdots \vee (E_m \wedge H(G_m, s)),$$

   for some goals $G_1, \ldots, G_m$, and equality constraints $E_1, \ldots, E_m$.
4. Choose an $1 \leq i \leq m$, and do the following. Let $\mathcal{G}$ be

   $$F_1(\vec{t_1}) \,\&\, \cdots \,\&\, F_n(\vec{t_n}) \,\&\, G_i.$$

   Let $\mathcal{E}$ be $\mathcal{E} \wedge E \wedge E_i$. Append $A(\vec{t})$ to Γ.
5. Go back to step 2.

Clearly, this planner is sound and complete: There is a plan for the goal G_0 iff there is an execution of this procedure that gives the plan as Γ. By Theorem 1, the following variant of this planner is also sound and complete:

[6] For any expression e and variable substitution σ, we use $e \cdot \sigma$ to denote the result of simultaneously repacing the free variables in e according to σ. Bound variables in e must be renamed in order to avoid the capture of free variables during the substitution process.

[7] Notice that this is always possible for context free action theories.

1. Initialize Γ (for partial plans) to $\emptyset$, $\mathcal{G}$ (for outstanding goals) to G_0, and $\mathcal{E}$ (for equality constraints) to $true$.
2. If there is a variable substitution σ such that $\mathcal{D}_{S_0} \cup \mathcal{D}_{una} \models (H(\mathcal{G}, S_0) \wedge \mathcal{E}) \cdot \sigma$, then output $\Gamma \cdot \sigma$ as the final plan.
3. Choose a simple goal g in $\mathcal{G}$ with the property that there is no other simple goal g' in $\mathcal{G}$ such that $\mathcal{D} \models (\forall \vec{x})\mathcal{E} \supset g \prec g'$ holds, where $\vec{x}$ is the tuple of free variables in g, g', and $\mathcal{E}$.
4. Choose an action $A(\vec{t})$ that has a positive effect on g. Suppose the axiom for $Poss(A(\vec{t}), s)$ is as (6):

   $$H(F_1(\vec{t_1}), s) \wedge \cdots \wedge H(F_n(\vec{t_n}), s) \wedge E,$$

   and suppose that under the unique names axioms $\mathcal{D}_{una}$, $\mathcal{R}[H(\mathcal{G}, do(A(\vec{t}), s))]$ is equivalent to:

   $$(E_1 \wedge H(G_1, s)) \vee \cdots \vee (E_m \wedge H(G_m, s)),$$

   for some goals $G_1, \ldots, G_m$, and equality constraints $E_1, \ldots, E_m$.
5. Choose an $1 \leq i \leq m$, and do the following. Let $\mathcal{G}$ be

   $$F_1(\vec{t_1}) \,\&\, \cdots \,\&\, F_n(\vec{t_n}) \,\&\, G_i.$$

   Let $\mathcal{E}$ be $\mathcal{E} \wedge E \wedge E_i$. Append $A(\vec{t})$ to Γ.
6. Go back to step 2.

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 $on(B, C) \prec on(A, B)$, the planner should try $on(A, B)$ 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 $clear(B)$, $clear(A)$, or $ontable(A)$, but not $on(B, C)$.

6.2 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 3 *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*

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

Proof: The theorem follows directly from Theorem 1. ■

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 $\mathcal{D}_{S_0} \models H(G, S_0)$. 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 coordinated 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 $H(g,s)$ is a conjunct of $Poss(\tau(w),s)$.
3. $w \neq$ `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 $H(g_1,s) \wedge \cdots \wedge H(g_n,s) \wedge E$, 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 (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

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

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 the subgoals in $Open(w)$, and G be

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

Nondeterministically do one of the following steps and then go back to (1). Fail if none of the following steps succeed.

(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 $\mathcal{D}_{S_0} \models H(G, S_0)$, 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.

Example 6.1 Let us now illustrate this planner using a machine shop scheduling domain adapted from (Smith and Peot [22]).

Fluents:

- $shaped(x)$: x is shaped.
- $drilled(x)$: x is drilled.
- $fastened(x, y)$: x is fastened to y.
- $free(x)$: x is not fastened to any other object.

Actions:

- $shape(x)$: 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 as follows:

Action precondition axioms:

$$Poss(shape(x), s) \equiv H(free(x), s),$$

$$Poss(drill(x), s) \equiv H(free(x), s),$$

$$Poss(bolt(x, y), s) \equiv H(drilled(x), s) \wedge H(drilled(y), s) \wedge x \neq y.$$

Successor state axioms:

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

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

$$Poss(a, s) \supset H(fastened(x, y), do(a, s)) \equiv$$

$$a = bolt(x,y) \vee a = bolt(y,x) \vee H(fastened(x,y),s),$$

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

Now suppose that the initial database $\mathcal{D}_{S_0}$ is the set of following sentences:

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

It can be verified that the following are state constraints:

$$s \geq S_0 \supset (\forall x).H(free(x),s) \equiv \neg(\exists y)H(fastened(x,y),s),$$
$$s \geq S_0 \supset (\forall x,y).H(fastened(x,y),s) \equiv H(fastened(y,x),s).$$

Using these two state constraints and Theorem 2, we can show that

$$(\forall x,y).x \neq y \supset$$
$$shaped(x) \prec fastened(x,y) \wedge shaped(y) \prec fastened(x,y) \wedge$$
$$drilled(x) \prec fastened(x,y) \wedge drilled(y) \prec fastened(x,y) \wedge$$
$$shaped(x) \prec drilled(x)$$

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 [22]), some non-trivial threat removal strategies are needed in order to avoid backtracking. ■

7 Some General Considerations

In general, computing $\prec$ requires induction, thus is intractable. However, since $\prec$ is defined context independently, the computation does not need to be done at run time. It is even possible for users to provide such knowledge. For instance, the subgoal orderings in both the blocks world and the machine shop examples are obvious (to the human designer).

It is also possible for users to include $\prec$ in their goal statements. For instance, to force the planner to generate a plan that achieves $on(B, C)$ before $on(A, B)$, we can use the goal

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

Formally, we can say that a plan Γ achieves this goal under the background theory $\mathcal{D}$ iff

$$\mathcal{D} \models H(on(B, C) \,\&\, on(A, B), do(\Gamma, S_0)) \wedge Precedence(on(B, C), on(A, B), do(\Gamma, S_0)).$$

This kind of goals is sometimes useful. For instance, it is sometimes easier, and often sufficient, to tell someone to "first pick up the child, then go to the supermarket" than to include enough axioms to make this deducible.

Our goal ordering can be extended along several dimensions. Recall that the macro $g_1 \prec g_2$ is defined in terms of $Precedence(g_1, g_2, s)$ by quantifying over all legal situations that achieve g_1 and g_2. Instead of the space of all possible plans, we can consider some smaller classes of plans. For instance, similar to (Cheng and Irani [4]), we can consider only those plans that also achieve some additional goals:

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

8 Related Work

The situation calculus has been used for planning ever since it's introduced (McCarthy [14], McCarthy and Hayes [15], Green [7]). However, to the best of our knowledge, this paper is the first attempt in using the situation calculus to formalize control knowledge in planning.

Although developed independently, it turned out that our ordering relation is closely related to but subtly different from that in (Etzioni [5]) and that in (Cheng and Irani [4]). Although we consider only the space of *legal* situations, Cheng and Irani ([4]) consider the space of all possible situations.[8] 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. This is because given an (illegal) situation s such that

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

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

Our subgoal ordering is also related to various threat removal and conflict resolution strategies in nonlinear planning (Smith and Peot [22], Yang [24], and others). It is possible that some provably correct threat removal strategies can be derived from our goal ordering, and vice versa. However, our goal ordering has the advantage that it is planner independent. As we have shown, it is applicable to both linear and nonlinear planners.

This work is also related to the work of Bacchus and Kabanza [1]. While we use the situation calculus, Bacchus and Kabanza use a temporal logic for expressing control information in planning. However, it seems that much of the control information discussed in this paper can also be represented using their temporal logic, and vise versa.

9 Concluding Remarks

We have applied the situation calculus to formalizing control information in planning by formulating a subgoal ordering. We have shown the usefulness of this ordering to both linear and nonlinear planners.

Information on subgoal 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

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

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

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

[8] It seems that Etzioni ([5]) has an implicit notion of legality built into his algorithms by using partial evaluation and state constraints.

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

We have also applied the situation calculus to formalizing control information in logic programming by giving a logical semantics to the cut operator, the chief search control operator in Prolog. For details, see Lin [10]

Acknowledgements

Much of this work was done while the author was with the Department of Computer Science at the University of Toronto, where the research was supported in part by grants to Hector Levesque and Raymond Reiter from the Government of Canada Institute for Robotics and Intelligent Systems, and from the National Science and Engineering Research Council of Canada. This work is also supported in part by grant DAG96/97.EG34 from the Hong Kong Government Research Grant Council.

The author would like to thank the other members of Toronto Cognitive Robotics group (Yves Lesperance, Hector Levesque, Daniel Marcu, Ray Reiter, and Richard Scherl) for helpful discussions related to the subjects of this paper.

References

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

[2] 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.

[3] D. Chapman. Planning for conjunctive goals. *Artificial Intelligence*, 32:333–377, 1987.

[4] 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.

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

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

[7] 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.

[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. Applications of the situation calculus to formalizing control and strategic information: the prolog cut operator. http://www.cs.ust.hk/~flin/, Submitted.

[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. *Artificial Intelligence*, 1997. To appear.

[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] N. J. Nilsson. *Principles of Articifial Intelligence*. Morgan Kaufmann, Los Altos, CA., 1980.

[17] E. P. Pednault. Synthesizing plans that contain actions with context-dependent effects. *Computational Intelligence*, 4:356–372, 1988.

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

[19] 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: Papers in Honor of John McCarthy*, pages 418–420. Academic Press, San Diego, CA, 1991.

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

[21] R. Reiter. On specifying database updates. *Journal of Logic Programming*, 25(1):53–91, 1995.

[22] 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.

[23] R. Waldinger. Achieving several goals simultaneously. In E. Elcock and D. Michie, editors, *Machine Intelligence*, pages 94–136. Ellis Horwood, Edinburgh, Scotland, 1977.

[24] Q. Yang. A theory of conflict resolution in planning. *Artificial Intelligence*, 58:361–392, 1992.