On Measuring Plan Quality (A Preliminary Report)

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

Abstract

By using an example from a robot navigating domain, we argue that to specify declaratively the behavior of an agent, we need to have a formal and explicit notion of "quality plans." To that end, we propose the following three domain independent measures of plan quality: a plan is said to be A-minimal if none of the actions in it can be deleted and have it continue to be a plan; it is said to be B-minimal if none of its segments can be deleted and have it continue to be a plan; and it is said to be C-minimal if none of its segments can be replaced by a single action and have it continue to be a plan. We show that given a plan, checking if it satisfies any of these three measures, and if not transforming it into one that does, can be done in polynomial numbers of plan tests. We also show how these three notions of plan minimality can be axiomatized naturally and straightforwardly in the situation calculus and, as a result, show their relationships with STRIPS triangle tables. Finally, we show their applications in the declarative specifications of agents' behaviours.

1 Introduction

Consider the classical SRI robot domain ([4]) where the robot can perform various chores such as delivering a bagel to someone in an office environment. Consider the requirement "close any doors that you open". There are many possible interpretations for this requirement. The easiest one is to take it as a "restore" goal of ([16]), which amounts to the constraint: if initially a door is closed, then by the end of the plan the door is still closed. But as Bacchus and Kabanza [2] note, this constraint is often too weak because it allows plans that leave some doors open longer than necessary.

To overcome this problem, Bacchus and Kabanza propose the following constraint: if the robot opens a door at any point of her plan, then at the next point in time, she must go through the door, and at the next point in time after that, she must close the door. While does the right thing for the version of the robot domain considered by Bacchus and Kabanza in [2][1], it does not work in general as it presupposes that the robot is able to close a door immediately after she has gone through it. For example, suppose that in order to open or close a door, the robot's hand must be empty. Then in order to go through an initially closed door while carrying an object, the robot will have to first find an appropriate place to deposit the object. Similarly, in order to close a door after going through it while carrying an object, she has to find a place to deposit the object as well. Thus, in a domain like this, an arbitrarily long time may elapse between the moment that the robot opens a door and the moment that she close it again.

This is where a formal notion of "high quality" plans can help: a more general solution along the line of Bacchus and Kabanza's is to require that after the robot has gone through an initially closed door, she must set out immediately to close it, that is, the next segment of the robot plan must be one that achieves "closed(door)" without any unnecessary actions, i.e. a "high quality" plan for "closed(door)".

There are many possible interpretations for what counts as a high quality plan. In this paper, we shall study three conceptually simple, computationally appealing, and domain independent ones. At the present time, we are not advancing any of them as the ultimate criteria of quality plans, if there is any such one, rather

[1] In this domain, the robot can always open and close a door as long as she is next to the door.

we are using them to illustrate the need for having an explicit and formal notion of quality plans.

This paper is organized as follows. In the next section, we lay out a general framework for studying plan quality. Then in section 3 we define three measures of plan quality and prove some results about their worst time complexity. In section 4 we axiomatize them in the situation calculus and relate them to regression ([10, 12]) and the STRIPS triangle tables ([3]). In section 5 we return to the robot domain and show how the aforementioned requirement about the statuses of doors can be axiomatized. In section 6, we discuss some related work, and finally in section 7, we make some concluding remarks.

2 Vocabularies and Notations

We assume the following vocabularies:

- A set $\mathcal{A}$ of actions.
- A set $\mathcal{G}$ of goals.
- The set $\mathcal{SA}$ of finite sequences of actions in $\mathcal{A}$. In particular, the empty sequence [] is in $\mathcal{SA}$.
- A binary predicate *Plan* on $\mathcal{SA} \times \mathcal{G}$. For any $\sigma \in \mathcal{SA}$ and any $g \in \mathcal{G}$, $Plan(\sigma, g)$, read "σ is a *plan* for g", means that the sequence of actions σ can be successfully executed in the (implicit) starting situation to achieve the goal g.

In the following, for any σ, γ and ξ in $\mathcal{SA}$, and any natural number n, we let $\sigma_{\xi}^{\gamma,n}$ to stand for the result of replacing the n-th subsequence γ in σ by ξ. If there are no such subsequence, then we let $\sigma_{\xi}^{\gamma,n}$ be σ. For example, using the Prolog list notation for finite sequences, we have

$$[a\ b\ c\ d]_{[]}^{[a],1} = [b\ c\ d],$$

$$[a\ b\ c\ d]_{[a\ a]}^{[a],2} = [a\ b\ c\ d],$$

$$[a\ b\ c\ d]_{[a]}^{[b\ c],1} = [a\ a\ d],$$

$$[a\ b\ c\ a\ b\ d]_{[b]}^{[a\ b],2} = [a\ b\ c\ b\ d]$$

where $a, b, c, d \in \mathcal{A}$, [] is the empty sequence.

3 Measuring plan quality

The usual measure on plan quality is based on the idea that the best plan is the shortest:[2]

> σ is an *optimal* plan for g if it is plan for g and there does not exist another plan γ for g such that the length of γ is smaller than that of σ.

In this paper we shall not concern ourself with optimal plans, for several reasons. Firstly, optimal plans are expensive to check and hard to generate. Secondly, optimality is a global notion. The information contained in a given suboptimal plan may have no help at all in "transforming it" into an optimal one: the problem is just the same as a fresh optimal plan generation problem. Computational issues aside, optimal plans are often not what we want for the purpose of specifying agents' behaviors, as we shall see in section 5.

Instead, in this paper we shall investigate the following three tractable and local measures of plan quality that are wearker than the notion of optimality:

1. A-minimality: a plan is A-minimal if none of the actions in it can be deleted and have it continue to be a plan.
2. B-minimality: a plan is B-minimal if none of its segments can be deleted and have it continue to be a plan.
3. C-minimality: a plan is C-minimal if none of its segments with lengths greater than 1 can be replaced by a single action and have it continue to be a plan.

Formally, we have:

Definition 1 *Let σ be a plan for g. We say σ is* A-minimal *if there do not exist an action $\alpha \in \mathcal{A}$ and a number n such that $\sigma_{[]}^{[\alpha],n}$ is not identical to σ but is a plan for g.*

We say σ is B-minimal *if there do not exist a sequence of actions $\gamma \in \mathcal{SA}$ and a number n such that $\sigma_{[]}^{\gamma,n}$ is not identical to σ but is a plan for g.*

We say σ is C-minimal *if there do not exist an action $\alpha \in \mathcal{A}$, a sequence of actions $\gamma \in \mathcal{SA}$ with length greater than 1, and a number n such that $\sigma_{[\alpha]}^{\gamma,n}$ is not identical to σ but is a plan for g.*

We want to point out that these three measures are by no means the only local and tractable (see below) ones. In particular, it is interesting to notice that C-minimality can be generalized in various ways. For instance, instead of $\sigma_{[\alpha]}^{\gamma,n}$, if we consider $\sigma_{\xi}^{\gamma,n}$ for all ξ whose length is less than that of γ, then C-minimality will be the same as optimality.

[2] Implicit in this definition is the assumption that each action have equal cost. This will also be the assumption that we will make in the rest of this paper. The extension to the more general case should be straightforward.

We now proceed to show some properties of these three minimality measures. The following proposition is immediate.

Proposition 1 *Let σ be a plan for g. If σ is B-minimal, then it is also A-minimal. If σ is C-minimal, then it is also B-minimal, provided the empty sequence is not a plan for g.*

It is not hard to see that there may be A-minimal but not B-minimal plans, and similarly there may be B-minimal but not C-minimal plans.

Let σ be a finite sequence of actions. We call testing if $Plan(\sigma, g)$ hold a *plan test* for σ with respect to g. For typical action specifications given by, for example ADL [11] or basic action theories (see below), if the starting situation is complete, then plan tests for a fixed goal can be done efficiently.

For all three versions of plan minimality, checking if a given plan is minimal, and if not, transforming it into a minimal one can be done in polynomial number of plan tests. Thus if plan tests are tractable, as we believe they often are, so will be the plan minimality tests.

Proposition 2 *Given a $\sigma \in \mathcal{SA}$ with length n and a goal g, testing if σ is an A-minimal plan for g can be done in $O(n)$ number of plan tests for action sequences whose lengths are no more than n.*

If σ is a plan for g but not an A-minimal one, then an A-minimal one can be constructed from σ in $O(n^2)$ number of plan tests for action sequences whose lengths are no more than n.

Proof: There are only n actions in σ, so testing if any of them can be deleted can be done in $O(n)$ number of plan tests. More precisely, consider the following Prolog-like rules:

```
aMinimal(X,[]) :- plan(X,g).
aMinimal(X, [a | Y]) :- append(X,Y,Z),
   not plan(Z,g), append(X,[a],X1),
   aMinimal(X1,Y).
```

where $append(X1, X2, X3)$ if the list $X3$ is the result of joining the lists $X1$ and $X2$. Clearly, σ is an A-minimal plan for g iff the query $aMinimal([], \sigma)$ returns "yes". The desired complexity result then follows from this program.[3]

If σ is not A-minmal, then we can repeatedly eliminate redundant actions one by one until none can be deleted anymore. Since there are only n actions, and finding one to delete takes $O(n)$ plan tests, so in the worst case, it takes $O(n^2)$ plan tests. Again, more precisely, consider the following Prolog-like rules:

```
aMinimal(X, [], X) :- plan(X,g).
aMinimal(X, [a | Y], Z) :- append(X,Y,Y1),
   plan(Y1,g), aMinimal([], Y1, Z).
aMinimal(X, [a | Y], Z) :- append(X,Y,Y1),
   not plan(Y1,g), append(X, [a], X1),
   aMinimal(X1, Y, Z).
```

Now given the query $aMinimal([], \sigma, X)$, the X returned, if any, will be an A-minimal plan for g. ■

Proposition 3 *Given a $\sigma \in \mathcal{SA}$ with length n and a goal g, testing if σ is a B-minimal plan for g can be done in $O(n^2)$ number of plan tests for action sequences whose lengths are no more than n.*

If σ is a plan for g but not a B-minimal one, then a B-minimal one can be constructed (from σ) in $O(n^3)$ number of plan tests for action sequences whose lengths are no more than n.

Proof: Similar to last proposition except now there are $O(n^2)$ segments to consider. ■

Proposition 4 *Let $\mathcal{A}$ be finite and of size m. Given a $\sigma \in \mathcal{SA}$ with length n and a goal g, testing if σ is a C-minimal plan for g can be done in $O(mn^2)$ number of plan tests for action sequences whose lengths are no more than n.*

If σ is a plan for g but not a C-minimal one, then a C-minimal one can be constructed (from σ) in $O(mn^3)$ number of plan tests for action sequences whose lengths are no more than n.

Proof: Given a plan σ of length n, there are $(n-2)^2$ segments with lengths greater than 1 to consider. For each of the segment, checking if it can be replaced by a single action takes $O(m)$ plan tests. So checking C-minimality takes $O(mn^2)$ plan tests.

If a plan is not C-minimal, then since finding a segment and an action to replace it takes $O(mn^2)$ plan tests, and we need to do such substitution at most n times because each substitution will decrease the length of the plan by at least 1, it takes at most $O(mn^3)$ plan tests to transform the plan into a C-minimal one. ■

These results are for the general case. For action theories that permit regressions, we can show some more

[3] Notice that *append* can be implemented in such a way (e.g. through pointers) that it takes only constant time.

concrete results. In this paper we shall consider action specifications that are given as *basic action theories* [8] in the situation calculus.

4 Plan quality in basic action theories

As far as the effects of actions are concerned, basic action theories have the same expressive power as ADL [11]. However, these action theories go well beyond being just a language for specifying the effects of actions. They are logical theories that can be used to reason about plans and planning processes. They can also be easily extended to include various domain constraints and control information [7, 6]. Before introducing basic action theories, we first briefly review the situation calculus, and define some necessary notations.

4.1 The situation calculus

Our version of the situation calculus [9] employs a many sorted second-order language. We assume the following sorts: *situation* for situations, *action* for actions, *fluent* for propositional fluents, and *object* for everything else.

We use the following domain independent predicates and functions:

- A distinct constant S_0 denoting the initial situation.
- A binary function do - for any action a and any situation s, $do(a, s)$ is the situation resulting from performing a in s.
- A binary predicate H - for any propositional fluent p and any situation s, $H(p, s)$ is true if p holds in s.
- A binary predicate $Poss$ - for any action a and any situation s, $Poss(a, s)$ is true if a is possible (executable) in s.
- A binary relation $\sqsubset$ on situations - for any situations s and s', $s \sqsubset s'$ if s' is the resulting situation of doing one or more actions in s. In the following, we consider $s \sqsubseteq s'$ to be a shorthand for $s \sqsubset s' \vee s = s'$.

We'll have a few other special predicate and functions, and will introduce them when they are needed.

We shall assume the following foundational axioms (cf. [14]):

$$S_0 \neq do(a, s), \tag{1}$$

$$do(a_1, s_1) = do(a_2, s_2) \supset (a_1 = a_2 \wedge s_1 = s_2), \tag{2}$$

$$(\forall P)[P(S_0) \wedge (\forall a, s)(P(s) \supset P(do(a, s))) \supset (\forall s)P(s)], \tag{3}$$

$$\neg s \sqsubset S_0, \tag{4}$$

$$s \sqsubset do(a, s') \equiv s \sqsubseteq s'. \tag{5}$$

The first two axioms are unique names assumptions about situations. 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 repeatedly applying the function do. The last two axioms define $\sqsubset$ inductively.

By our foundational axioms, for any situation s, there must be a unique finite sequence of actions σ such that $s = do(\sigma, S_0)$, where for any s', any action a, and any sequence L, we define:[4]

$$do([], s') \stackrel{def}{=} s', \quad do([a|L], s') \stackrel{def}{=} do(L, do(a, s')).$$

This means that by our foundational axioms, the set of situations is isomorphic to the set of finite sequences of action. This isomorphism will be the logical basis on which properties of plans are axiomatized in the situation calculus.

4.2 Fluent formulas

We will find it convenient to talk about *fluent formulas*, which are generalizations of those in [15]. Informally, a fluent formula is a situation calculus formula with situation terms suppressed. Formally, we define them as follows:

1. Any formula that does not mention a situation term is a fluent formula.
2. If F is a n-ary fluent, and $t_1, ..., t_n$ are object terms, then $F(t_1, ..., t_n)$ is a fluent formula.
3. If ϕ and φ are fluent formulas, x an object variable, and p a fluent variable, then $\neg\varphi$, $\varphi \vee \phi$, $(\forall x).\varphi$ and $(\forall p).\phi$ are fluent formulas. (Other connectives such as $\wedge$, $\supset$, $\equiv$, and $\exists$ can be defined as usual.)

For example, $A1 \neq A2$, $on(block1, block2)$, $(\forall p)(p \supset p)$ are fluent formulas, where $A1$ and A are action constants and on a binary fluent.

[4] We write $e1 \stackrel{def}{=} e2$ if $e1$ is considered to be a shorthand for $e2$.

Given a fluent formula φ and a situation term S, we define $H(\varphi, S)$ to be a shorthand for a situation calculus formula according to the following rules:

$$
\begin{aligned}
&H(\varphi, S) \stackrel{def}{=} \varphi, \text{ provided } \varphi \text{ does not mention} \\
&\qquad\qquad\qquad \text{any situation terms} \\
&H(\neg\varphi, S) \stackrel{def}{=} \neg H(\varphi, S), \\
&H(\varphi \vee \phi, S) \stackrel{def}{=} H(\varphi, S) \vee H(\phi, S), \\
&H((\forall x)\varphi, S) \stackrel{def}{=} (\forall x)H(\varphi, S), \\
&H((\forall p)\varphi, S) \stackrel{def}{=} (\forall p)H(\varphi, S).
\end{aligned}
$$

For example,

$$
\begin{aligned}
&H(A_1 \neq A_2, s) \stackrel{def}{=} A_1 \neq A_2, \\
&H(on(B1, B2) \vee clear(B2), s) \stackrel{def}{=} \\
&\quad H(on(B1, B2), s) \vee H(clear(B2), s), \\
&H((\forall x)[clear(x) \equiv \neg(\exists y)on(y, x)], s) \stackrel{def}{=} \\
&\quad (\forall x)[H(clear(x), s) \equiv \neg(\exists y)H(on(y, x), s)].
\end{aligned}
$$

4.3 Plans and minimal plans

We now proceed to axiomatize the three versions of plan minimality in the situation calculus.

First of all, we take a *goal* to be a ground fluent formula. Secondly, instead of finite sequences of actions, we consider situations. Because of the isomorphism between situations and finite sequences of actions that we mentioned above, we don't lose anything by doing this. This change of ontology is important here because situations are first class objects in the situation calculus, while finite sequences of actions are not. Finally, we make explicit the starting situation in which a plan is to be executed. Therefore, instead of a binary $Plan$ predicate, we extend it to a ternary one: for any goal g, situations s_1 and s_2, $Plan(g, s_1, s_2)$ if the sequence of actions that takes s_1 (the starting situation) to s_2 (the goal situation) is executable in s_1 and makes g true afterward:

$$
\begin{aligned}
&Plan(g, s_1, s_2) \stackrel{def}{=} H(g, s_2) \wedge s_1 \sqsubseteq s_2 \wedge \\
&\quad (\forall a, s)[s_1 \sqsubset do(a, s) \sqsubseteq s_2 \supset Poss(a, s)].
\end{aligned}
$$

Corresponding to the three versions of plan minimality defined earlier, we have the following three macros:

$$
\begin{aligned}
&Aminimal(g, s_1, s_2) \stackrel{def}{=} Plan(g, s_1, s_2) \wedge \\
&\quad (\forall a, s)[s_1 \sqsubset do(a, s) \sqsubseteq s_2 \supset \\
&\qquad \neg Plan(g, s_1, del(s_2, s, do(a, s)))],
\end{aligned}
$$

$$
\begin{aligned}
&Bminimal(g, s_1, s_2) \stackrel{def}{=} Plan(g, s_1, s_2) \wedge \\
&\quad (\forall s', s'')[s_1 \sqsubseteq s' \sqsubset s'' \sqsubseteq s_2 \supset \\
&\qquad \neg Plan(g, s_1, del(s_2, s', s''))], \\
&Cminimal(g, s_1, s_2) \stackrel{def}{=} Plan(g, s_1, s_2) \wedge \\
&\quad (\forall a, s', s'', s^*)[s_1 \sqsubseteq s' \sqsubset s^* \sqsubset s'' \sqsubseteq s_2 \supset \\
&\qquad \neg Plan(g, s_1, repl(s_2, s', s'', a))],
\end{aligned}
$$

where for any situations s, s_1, s_2, $del(s, s_1, s_2)$, the situation resulted from deleting the segment of s between s_1 and s_2, is the unique situation that satisfies the following axioms:

$$\neg(s_1 \sqsubseteq s_2 \sqsubseteq s) \supset del(s, s_1, s_2) = s, \quad (6)$$

$$s_1 \sqsubseteq s_2 \supset del(s_2, s_1, s_2) = s_1, \quad (7)$$

$$
\begin{aligned}
&s_1 \sqsubseteq s_2 \sqsubset do(a, s) \supset \\
&\quad del(do(a, s), s_1, s_2) = do(a, del(s, s_1, s_2)), \quad (8)
\end{aligned}
$$

and for any situations s, s_1, s_2, and any action a, $repl(s, s_1, s_2, a)$, the result of replacing the segment between s_1 and s_2 in s by the action a, is the unique situation that satisfies the following axioms:

$$\neg(s_1 \sqsubset s_2 \sqsubseteq s) \supset repl(s, s_1, s_2, a) = s, \quad (9)$$

$$s_1 \sqsubset s_2 \supset repl(s_2, s_1, s_2, a) = do(a, s_1), \quad (10)$$

$$
\begin{aligned}
&\quad s_1 \sqsubset s_2 \sqsubset do(b, s) \supset \\
&repl(do(b, s), s_1, s_2, a) = do(b, repl(s, s_1, s_2, a)). \quad (11)
\end{aligned}
$$

In the following, we shall denote by $\mathcal{D}_{min}$ the set of axioms (6) – (11).

4.4 Basic action theories

A *basic action theory* $\mathcal{D}$ is one that has the following form: (cf. Reiter [13] and Lin and Reiter [8]):

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

where

- Σ is the set of the foundational axioms (1) – (5).
- $\mathcal{D}_{ss}$ is a set of successor state axioms, one for each fluent F, of the form:

  $$Poss(a, s) \supset H(F(\vec{x}), do(a, s)) \equiv H(\Phi_F(\vec{x}, a), s), \quad (12)$$

  where $\Phi_F(\vec{x}, a)$ is a fluent formula whose free variables are among $\vec{x}, a$. Intuitively, a successor state axiom for F defines the truth value of F in a successor situation in terms of the truth values of fluents in the current situation.
- $\mathcal{D}_{ap}$ is a set of action precondition axioms of the form:

  $$Poss(A(\vec{x}), s) \equiv H(\Psi_A(\vec{x}), s), \quad (13)$$

where A is an action, and $\Psi_A(\vec{x})$ is a fluent formula whose free variables are among $\vec{x}$.

- $\mathcal{D}_{una}$ is the set of unique names axioms for actions.
- $\mathcal{D}_{S_0}$, the initial database, is a finite set of first-order sentences of the form $H(\phi, S_0)$, where ϕ is a ground fluent formula.

Basic action theories are computationally appealing because they permit reasoning by regression:

Definition 2 (Regression) *Let $\mathcal{D}$ be a basic action theory. Let g be a fluent formula, and α an action term of the form $A(\vec{t})$. The regression of g over α (under $\mathcal{D}$), written $R(g, \alpha)$, is the fluent formula obtained from g by replacing every expression of the form $F(\vec{u})$ in it by $\Phi_F(\vec{u}, A(\vec{t}))$, where Φ_F is as in the successor state axiom (12) for F.*

The following is the fundamental property of regression:

Proposition 5 *Let $\mathcal{D}$ be a basic action theory, then*

$$\mathcal{D} \models (\forall s).Poss(\alpha, s) \supset H(g, do(\alpha, s)) \equiv H(R(g, \alpha), s).$$

The proof of a proposition similar to this one can be found in (Reiter [14]).

4.5 Minimal plans under basic action theories

We now show how to check if a given plan is a minimal one using regression under a basic action theory.

Given a basic action theory $\mathcal{D}$, a goal g, and a finite sequence of actions $[\alpha_1, ..., \alpha_n]$, let

$$g_{n+1} = g,$$
$$g_k = R(g_{k+1}, \alpha_k) \wedge \Psi_A(\vec{t}), \text{ for } 1 \le k \le n$$

where $\alpha_k = A(\vec{t})$ for some action A and tuple $\vec{t}$, and Ψ_A is as in (13) in $\mathcal{D}$.

The intuition behind g_k is that it is the condition under which the sequence of actions $[\alpha_k, ..., \alpha_n]$ can be successfully executed to achieve the goal g:

Lemma 1 *Under the basic action theory $\mathcal{D}$, we have:*

$$(\forall s).H(g_k, s) \equiv Plan(g, s, do([\alpha_k, ..., \alpha_n], s)).$$

Proof: By the definition of $Plan$, we need to prove the following equivalence under $\mathcal{D}$:

$$(\forall s).H(g_k, s) \equiv H(g, do([\alpha_k, ..., \alpha_n], s)) \wedge Poss([\alpha_k, ..., \alpha_n], s),$$

where $Poss([\alpha_k, ..., \alpha_n], s)$ is defined as follows:

$$Poss([], s) \stackrel{def}{=} true,$$
$$Poss([\beta, \beta_1, ..., \beta_m], s) \stackrel{def}{=} Poss([\beta_1, ..., \beta_m], do(\beta, s)) \wedge Poss(\beta, s).$$

We prove this by induction on k, with $k = n + 1$ as the base case, which is trivial:

$$(\forall s).H(g_{n+1}, s) \equiv H(g, do([], s)) \wedge Poss([], s).$$

Inductively, assume that we have

$$(\forall s).H(g_k, s) \equiv H(g, do([\alpha_k, ..., \alpha_n], s)) \wedge Poss([\alpha_k, ..., \alpha_n], s),$$

for $1 \le k = m \le n + 1$. We show that this is the case for $k = m - 1$ as well. Assuming that α_{m-1} is $A(\vec{t})$, we have

$$\begin{aligned}
&H(g_{m-1}, s) \equiv \\
&H(\Psi_A(\vec{t}), s) \wedge H(R(g_m, \alpha_{m-1}), s) \equiv \\
&Poss(\alpha_{m-1}, s) \wedge H(R(g_m, \alpha_{m-1}), s) \equiv \\
&\quad \text{by the basic property of regression} \\
&Poss(\alpha_{m-1}, s) \wedge H(g_m, do(\alpha_{m-1}, s)) \equiv \\
&\quad \text{by the inductive assumption} \\
&Poss(\alpha_{m-1}, s) \wedge Poss([\alpha_m, ..., \alpha_n], do(\alpha_{m-1}, s)) \wedge \\
&\qquad H(g, do([\alpha_m, ..., \alpha_n], do(\alpha_{m-1}, s))) \equiv \\
&Poss([\alpha_{m-1}, ..., \alpha_n], s) \wedge H(g, do([\alpha_{m-1}, ..., \alpha_n], s)).
\end{aligned}$$

This proves the inductive step, thus the lemma. ■

The sequence of fluent formulas $g_1, ..., g_{n+1}$ captures the essence of the triangle table [3] of a plan: g_i is essentially the i-th kernel of the plan. Just as STRIPS uses kernels to control plan execution, we can use these fluent formulas to check various notions of plan minimality. We show here how this can be done for B-minimality. Similar results hold for A-minimality and C-minimality.

Theorem 1 *Let $\mathcal{D}$ be a basic action, g a goal, and $[\alpha_1, ..., \alpha_n]$ a finite sequence of ground action terms. Under $\mathcal{D} \cup \mathcal{D}_{min}$, we have:*

$$\begin{aligned}
&Plan(g, s, do([\alpha_1, ..., \alpha_n], s)) \supset \\
&\quad Bminimal(g, s, do([\alpha_1, ..., \alpha_n], s)) \equiv \\
&\qquad \{ \bigwedge_{0 \le k < n,\, k+2 \le j \le n+1} \neg H(g_j, do([\alpha_1, ..., \alpha_k], s)) \}.
\end{aligned}$$

Proof: It is more convenient to prove the following equivalent sentence:

$$Plan(g,s,do([\alpha_1,...,\alpha_n],s)) \supset$$
$$\neg Bminimal(g,s,do([\alpha_1,...,\alpha_n],s)) \equiv$$
$$\{ \bigvee_{0\le k<n,\, k+2\le j\le n+1} H(g_j, do([\alpha_1,...,\alpha_k],s))\}.$$

Suppose $Plan(g,s,do([\alpha_1,...,\alpha_n],s))$.

$\Rightarrow$:

Assume

$$\neg Bminimal(g,s,do([\alpha_1,...,\alpha_n],s)).$$

Then by the definition of *Bminimal*, there must be two situations s' and s'' such that

$$s \sqsubseteq s' \sqsubset s'' \sqsubseteq do([\alpha_1,...,\alpha_n],s)$$

and

$$Plan(g,s,del(do([\alpha_1,...,\alpha_n],s),s',s'')).$$

By our foundational axioms in Σ, there must be $0 \le i < j \le n$ such that

$$s' = do([\alpha_1,...,\alpha_i],s),\ \ s'' = do([\alpha_1,...,\alpha_j],s)$$

By $\mathcal{D}_{min}$, this means that

$$del(do([\alpha_1,...,\alpha_n],s),s',s'')) =$$
$$do([\alpha_1,...,\alpha_i,\alpha_{j+1},...,\alpha_n],s).$$

Therefore

$$Plan(g,s,do([\alpha_1,...,\alpha_i,\alpha_{j+1},...,\alpha_n],s))$$

Thus

$$Plan(g,s_i,do([\alpha_{j+1},...,\alpha_n],s_i))$$

where

$$s_i = do([\alpha_1,...,\alpha_i],s).$$

So by Lemma 1, we have $H(g_{j+1},s_i)$. This completes the proof of "$\Rightarrow$".

$\Leftarrow$:

Assume that for some $0 \le k < n$, and $k+2 \le j \le n+1$, we have

$$H(g_j, s_k),$$

where $s_k = do([\alpha_1,...,\alpha_k],s)$. By Lemma 1, we have

$$Plan(g,s_k,do([\alpha_j,...,\alpha_n],s_k)).$$

Together with $Plan(g,s,do([\alpha_1,...,\alpha_n],s))$, we have

$$Plan(g,s,do([\alpha_1,...,\alpha_k,\alpha_j,...,\alpha_n],s).$$

Since

$$del(do([\alpha_1,...,\alpha_n],s),s_k,s'')) =$$
$$do([\alpha_1,...,\alpha_k,\alpha_j,...,\alpha_n],s),$$

where

$$s'' = do([\alpha_1,...,\alpha_{j-1}],s),$$

we conclude that $\neg Bminimal(g,s,do([\alpha_1,...,\alpha_n],s))$. This completes the proof of "$\Leftarrow$". ∎

5 The door please, Shakey

To show the usefulness of a logical formalization of plan quality, we now return to the robot domain that we discussed earlier in the paper. Our aim is to make precise the requirement "close those doors that you open." As we mentioned, there are several ways to interpret this requirement. We start with the easiest, taking it as a "restore" goal of [16]:

$$(\forall d).H(closed(d),S_0) \supset H(closed(d),S_{end}), \qquad (14)$$

where $S_{end} = do(\sigma, S_0)$, σ the given plan, and $closed(d)$ a fluent meaning that door d is closed.

Axiom (14) says that if a door is initially closed, then after the plan is done, it must still be closed.[5] As we discussed earlier, this constraint is often too weak as it allows plans that leave a door open longer than necessary, even if we insist on σ being, for example, an optimal plan.

There are several ways to strengthen this constraint. One is to require the robot to immediately set out to close an initially closed door right after *the last time* she went through it:

$$\begin{aligned}&(\forall d,s).\{H(closed(d),S_0) \wedge do(\text{go-thru}(d),s) \sqsubseteq S_{end} \wedge\\ &\quad (\forall s')[do(\text{go-thru}(d),s') \sqsubseteq S_{end} \supset s' \sqsubset s]\} \supset \qquad (15)\\ &\qquad (\exists s_1)(s \sqsubseteq s_1 \sqsubseteq S_{end} \wedge Bminimal(closed(d),s,s_1)).\end{aligned}$$

here go-thru(d) is the action that the robot performs to successfully go through door d. Notice that we use *Bminimal* in the above axiom to stand for "quality plans".

Notice that axiom (15) requires the robot to close a door as soon as possible the last time she goes through it, but leaves open what to do with the door at other times. To be more stringent, a user may require that every time the robot go through an initially closed door, she has to close it as soon as possible:

$$\begin{aligned}&(\forall d,s).\{H(closed(d),S_0) \wedge do(\text{go-thru}(d),s) \sqsubseteq S_{end}\} \supset\\ &\quad (\exists s_1)(s \sqsubseteq s_1 \sqsubseteq S_{end} \wedge Bminimal(closed(d),s,s_1)).\end{aligned} \qquad (16)$$

Constraint (16) is a generalization of that in [2] except that we do not require the robot to go through a door as soon as she opens it. To do that, we would have to axiomatize why the robot needs to go through the

[5] One should look at this axiom in conjunction with possibly other constraints such as the executability of the plan σ.

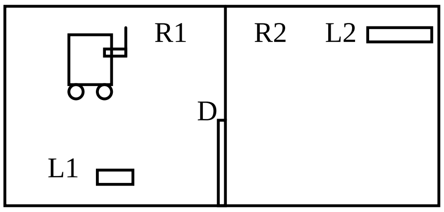

Figure 1: Shakey's world

door in the first place, which is, while something worth doing, beyond the scope of the present paper.[6]

Notice that in (15) and (16), insteard of *Bminimal*, one can use *Aminimal*, *Cminimal*, or any other measure one might have to stand for "quality plans". In general, the stronger the measure, the tighter the resulting constraint will be. However, one needs to be careful here not to use a measure that is too strong as to make the constraint unsatisfiable. In particular, one should not insist on using optimal plans here. For instance, consider the scenario as depicted in Figure 1:

- initially the robot is in room R1 and she needs to deliver the object that she's holding to location L2 in room R2;
- the rooms R1 and R2 are connected by door D;
- the robot's hand needs to be empty in order to close a door;
- at door D, the nearest place that the robot can put down an object is location L1 in room R1.

It can be seen that for a scenario like this, there is no plan σ that will achieve the goal of having the object at location $L2$ and satisfy at the same time the constraint (16) with *Bminimal* replaced by a formula corresponding to optimal plans: in order to open the door, the robot needs to put down the object that she's holding first; after she puts down the object and opens the door, she then needs to go back and pick up the object again because that's what she is trying to delivering; after she picks up the object and carries it through the door, she will now have to find an optimal plan to close the door, which is to put down the object in location $L1$ and go to the door to close it; but then she needs to open the door again in order to go to room $R2$, thus enters into a vicious cycle.

[6] For instance, if the robot is trying to carry an object to another room, then after she opens the door, the next goal should be going through the door to the other room *while carrying the desired object.*

This is the reason that we alluded to earlier in section 3 when we argued for the need of suboptimal plans.

Of course, a formal specification of desirable behaviours is only the first step. More difficult questions concern formal properties of the specifications and methods to generate plans that satisfy them. This, we believe, is where the advantage of working under the situation calculus, in particular basic action theories, comes in. For instance, for the robot domain, it can be shown that if there is a plan that satisfies (14), then there is also a plan that satisfies (15).

6 Related work

To the best of our knowledge, the most closely related work is that by Foulser, Li, and Yang [5]. They consider what they call the plan merging problem: given that "knowledge is available about what operators can be merged, and for each set of these operators, what the merged operators are... finding and analyzing methods for computing the optimal plan" (page 156, [5]). On the one hand, our work complements that in [5] in that we built into our definitions what operators can be "merged." For instance, according to B-minimality, sets of mergeable operators are identified with linear segments of the given plan, and the merged operator is the empty sequence (or the idle "wait" operator). On the other hand, our emphasis is different than that in [5]. Given a plan σ, while Foulser, Li, and Yang would look for the optimal, e.g. B-minimal "subplan" of σ, we are only concerned with obtaining a B-minimal subplan, i.e., for us, one B-minimal plan is as good as any other ones.

Our work is also related to that by Ambite and Knoblock [1] who propose a planning strategy that divides the planning process into two steps: initial plan generation, and plan refinement. While much of [1] is about domain specific rules that would transform a given plan into a "better" one, we are mainly interested in domain independent measures here. It would be an interesting future research topic to see how useful a domain independent notion of plan quality such as one of those studied here would be to a planning strategy such as Ambite and Knoblock's.

7 Concluding remarks

We have argued for the need of a formal notion of plan quality other than the usual notion of optimal (shortest) plans. In particular, we have proposed three local, tractable, and increasingly restrictive measures of plan quality, and studied their properties. We have provided a logical formalization for them in the situation calculus, show their relationships with regression and STRIPS triangle tables, and explore their possible applications in the formal specifications of robots' behaviours.

This work is part of our project on using the situation calculus to formalize control knowledge in planning, and is motivated primarily by the robot navigating example discussed above. There are many possible directions for future work. One of them is to use the situation calculus to provide a full declarative specification of the robot's desirable behaviors in the navigating domain.

Acknowledgements

I would like to thank the two anonymous KR'98 reviewers for their useful comments about an earlier version of this paper. This research was supported in part by the Research Grants Council of Hong Kong under Competitive Earmarked Research Grant HKUST6091/97E.

References

[1] J. L. Ambite and C. A. Knoblock. Planning by rewriting: efficiently generating high-quality plans. In *Proceedings of the 14th National Conference on Artificial Intelligence (AAAI–97), AAAI Press, Menlo Park, CA.*, pages 706–713, 1997.

[2] 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.

[3] R. E. Fikes, P. E. Hart, and N. J. Nilsson. Learning and executing generalized robot plans. *Artificial Intelligence*, 3(4):251–288, 1972.

[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] D. E. Foulser, M. Li, and Q. Yang. Theory and algorithms for plan merging. *Artificial Intelligence*, 57:143–181, 1992.

[6] F. Lin. Applications of the situation calculus to formalizing control and strategic information: the prolog cut operator. In *Proceedings of the Fifteen International Joint Conference on Artificial Intelligence (IJCAI–97), IJCAI Inc. Distributed by Morgan Kaufmann, San Mateo, CA.*, pages 1412–1418, 1997.

[7] F. Lin. An ordering on subgoals for planning. *Annals of Mathematics and Artificial Intelligence*, 21:321–342, 1997.

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

[9] 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.

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

[11] E. P. Pednault. ADL: Exploring the middle ground between STRIPS and the situation calculus. In *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning (KR'89)*, pages 324–332. Morgan Kaufmann Publishers, Inc., 1989.

[12] 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.

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

[14] R. Reiter. *Knowledge in Action: Logical Foundations for Describing and Implementing Dynamic Systems.* In preparation, 1997.

[15] H. Turner. A logic of universal causation. *Artificial Intelligence*, 1998. To appear.

[16] D. Weld and O. Etzioni. The first law of robotics (a call to arms). In *Proceedings of the 12th National Conference on Artificial Intelligence (AAAI–94), AAAI Press, Menlo Park, CA.*, pages 1042–1047, 1994.