MITSUBISHI ELECTRIC RESEARCH LABORATORIES
http://www.merl.com

Continuous Plan Evaluation with Incomplete Action Descriptions

Andrew Garland and Neal Lesh

TR2002-26 December 2002

Abstract

In complex environments, a planning system may be faced with incomplete information about both the planning operators and the state of the world. Previous research has investigated policies for selecting the best plan to execute, given incomplete operator descriptions. This paper extends that work to support interleaved planning and execution by taking into account past and current sensor readings.

3rd Intl NASA workshop on Planning and Scheduling for Space

Submitted June 2002.
Revised September 2002.

Continuous Plan Evaluation with Incomplete Action Descriptions

Andrew Garland and Neal Lesh
Mitsubishi Electric Research Laboratories
{garland,lesh}@merl.com

Abstract

In complex environments, a planning system may be faced with incomplete information about both the planning operators and the state of the world. Previous research has investigated policies for selecting the best plan to execute, given incomplete operator descriptions. This paper extends that work to support interleaved planning and execution by taking into account past and current sensor readings.

1 Introduction

In complex environments, a planning system may be faced with incomplete information about both the planning operators and the state of the world. While there has been substantial research effort devoted to planning in the face of incomplete state information (e.g, Levesque, 1996; Golden, 1998; Babaian & Schmolze, 2000), there has been little research on how to plan with incomplete planning operators, i.e., operators that may be missing preconditions and effects. Most past work in this area has produced machine learning techniques for improving operator descriptions for future planning episodes (e.g., Gil, 1994; Wang, 1995; Oates & Cohen, 1996). Garland & Lesh (2002) reported on an initial exploration into a complementary approach that focuses on selecting the best plan to execute for the current goal. This paper extends that work to address issues pertinent to continuous planning and re-planning.

Our work is partly motivated by the premise that complete action descriptions are sometimes impossible, and always difficult and time-consuming, to construct. For example, action descriptions cannot be guaranteed to be complete for domains that human experts cannot directly experience, such as navigating on the surface of another planet. In our approach, the domain experts who generate the action descriptions can provide additional information about the completeness of the model, similar to statements used to reason about incomplete states. For example, the experts can indicate which actions have been completely described, or that executing an action will not change the truth value of a domain literal.

The advantage of the approach proposed in Garland & Lesh (2002) is to more often execute plans that achieve their goals. For example, the techniques will sometimes prefer a plan because of certain orderings of actions, or the substitution of an action by one with similar effects. Even in the case where the domain experts provide no information about the completeness of the model, the techniques can select plans that are more likely to succeed than if the incompleteness of the model is ignored.

This paper extends Garland & Lesh (2002) by generalizing the techniques to support interleaved planning and execution based on (simplified models of) sensors and observed external events. The extensions are equally useful for continuous planning sessions that involve solving a long series of goals. The techniques we present reason efficiently over arbitrarily long execution histories by integrating the anticipated effects of past actions together with past sensor readings to form a model of the current state of the world.

2 Background

This section describes the framework and representations developed by Garland & Lesh (2002); our extensions are described in Section 3.

We assume a traditional planning representation. We also assume that for any planning domain, there exists a set of complete and correct action descriptions, which we will refer to as D_{true}. The set of action descriptions D available to a planner (or plan evaluator) will be a subset of the information in D_{true}.

Each action description consists of a specification of its preconditions and effects as sets of fluent literals. Thus, each action description a in D may include only a subset of a's preconditions and effects in D_{true}. In future work, we will look to extend the representations and techniques in this work to accommodate action descriptions that include conditional effects.

A plan is a sequence of actions that is intended to achieve a goal when executed in an initial state. Executing an action in a state in which all of its preconditions are true will produce a new state in which all of its effects are true and all other fluents remain unchanged. Executing an action with any false preconditions has no anticipated effect on the state. We define $achieves(D,S,g,p)$ as returning true iff the goal g is true in the state that results from simulating the execution of plan p from state S assuming that D is correct and complete.

D may be supplemented with statements about the completeness of action descriptions. Our approach follows the use of locally closed-world (LCW) statements that are used

Figure 1: **Action order.** These two plans differ only in the order of the actions, but they have different numbers of risks.

to overcome incomplete state information. In that setting, an LCW statement expresses limited knowledge of the state, such as that all the files that exist in a particular directory are known by a software agent (Golden, 1998; Babaian & Schmolze, 2000). In our work, an LCW statement expresses limited completeness of action descriptions.

LCW statements about actions in D are defined in terms of three predicates: $DoesNotRelyOn$, $DoesNotMakeTrue$, and $DoesNotMakeFalse$.[1] The statement $DoesNotMakeTrue(a, x)$ asserts that a does not have effect x in D_{true}, where x is a literal. The statement $DoesNotMakeFalse(a, x)$ asserts that a does not have effect $\neg x$ in D_{true}. The statement $DoesNotRelyOn(a, x)$ asserts that action a does not have precondition x or $\neg x$ in D_{true}. We define $CompletePreconditions(a)$ as:

$$\forall x. x \notin preconditions(a) \supset DoesNotRelyOn(a, x)$$

2.1 Plan selection based on risk assessment

The aim of this work is to allow a planning system to make better-informed decisions when deciding what plan to execute. More precisely, a *plan selection problem* is a 5-tuple (g, C, S, D, L), where g is a goal, C is a set of candidate plans, S describes the current state, D is a potentially incomplete action model, and L is a set of locally closed world statements. The ideal solution to this problem is to find the plan that will achieve its goal given the actual action descriptions, i.e. to find $c_i \in C$ so that $achieves(D_{true}, S, g, c_i)$. The quality of a plan selection algorithm can be measured by how frequently it chooses a plan that actually achieves g when executed in the current state.

The next section presents a tractable algorithm for identifying the *risks* of a plan, each of which represents a potential source of execution failure due to incompleteness of the action model. Garland & Lesh (2002) defined the following four types of risk for a plan composed of actions $a_1, ..., a_n$.

- POSSCLOB(a_i, x) :: action a_i might have the effect $\neg x$, and there exists action a_j, for $i < j$ that has precondition x in D and no action between a_i and a_j has effect x in D.
- PRECOPEN(a_i) :: action a_i might have an unlisted precondition that will not be true when a_i is executed in $a_0, .., a_n$.
- PRECFALSE(a_i, x) :: a_i has a precondition x in D which will be false when executed in $a_0, .., a_n$, according to D.
- UNLISTEDEFFECT(a_i, x) :: the correctness of the plan relies on an effect x of a_i that is not listed in D, but might be part of D_{true}. This means that x is consistent with the description of a_i in D, but there is no evidence to support the hypothesis that x is part of the description of a_i in D_{true}.

The first two types of risks correspond to identifying when the plan relies on the fact that D is a good approximation to D_{true}. In contrast, the latter two risks rely on the incompleteness of the model in order to justify selecting plans that would fail if $D = D_{true}$. (It is worth considering plans with such risks in the case when none of the plans that achieve g is justified by the action descriptions in D.)

Garland & Lesh (2002) presented a variety of *plan selection policies*. Each policy takes two candidate plans, c_1 and c_2, as input and returns true if c_1 is preferred to c_2 based on their risk sets. One such policy that will be used in the examples of this paper is:

- $RP_w(c_1, c_2)$ :: $weighted(c_1) < weighted(c_2)$ where $weighted(c)$ returns a real number by adding together the number of each type of risk multiplied by a pre-defined weight for that risk type.

Risk assessment can be incorporated with other methods for preferring plans. For example, most planners have an implicit preference for selecting the shortest plan that achieves the goal. See (Garland & Lesh, 2002) for a discussion of how to integrate risk assessment with other plan-preference metrics.

Figure 1 shows a simple example designed to illustrate how our risk analysis can prefer one ordering of plan actions over another. In this figure, actions are surrounded by brackets, with the action's preconditions on the lower left, and the action's effects on the upper right.

Figure 1 shows a goal that can be achieved by executing two actions, a_1 and a_2, in either order. If the action model is complete then both plans will achieve their goal. Either plan can fail, however, if the model is incomplete. Candidate plan $C_1 = [a_1, a_2]$ could fail if a_2 has an effect $\neg r$ which clobbers a_1's effect. Similarly, plan $C_2 = [a_2, a_1]$ could fail if a_1 clobbers a_2's effect. However, C_2 could also fail if a_2 has effect $\neg p$ which would clobber a_1's known precondition. Thus, C_2 has more POSSCLOB risks than C_1 and RP_w would prefer C_1 to C_2.

[1] The truth value of these predicates is directly specified by the user; no inference is performed.

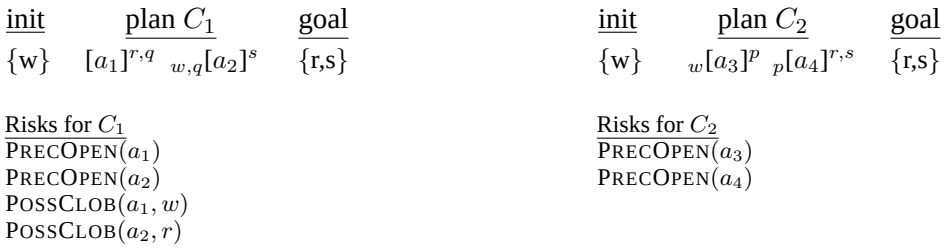

Figure 2: **Operator choice.** C_2 is preferred because C_1 has more possible clobberings.

Run	Risks percentile				Avg # of risks	Dev. of risks	Min # of risks	Max # of risks	Number of plans
	t_{10}	t_{20}	t_{50}	t_{100}					
Action ordering	87.6	86.2	83.1	76.7	49.4	4.3	42.0	74.0	195,254
Operator choice	85.4	83.0	79.8	74.7	72.0	7.8	55.0	115.0	341,469

Table 1: Impact of risk assessment on likelihood of successfully executing plans.

The preference for C_1 over C_2 is justified by imagining what we could add to D in order to create a D_{true} in which the plans would fail. For any combination of additional conditions and effects that would cause C_2 to fail, there is a corresponding modification that would cause C_1 to fail. However, in order to "match" the risk introduced by adding $\neg p$ as an effect of C_2, we have to add both a precondition and an effect to C_1. Thus, in the absence of any LCW information, C_1 seems the safer choice. On the other hand, if the given LCW eliminates the risks of plan C_2, then it becomes the better plan to execute.

There is another class of problems that shows the benefit of reasoning about the completeness of action descriptions. In these problems, the plans being compared contain alternative actions with similar effects. The most obvious role of our techniques would be to prefer to use operators that are completely modeled over ones that are not. In general, though, the issues that arise in the choice of actions to achieve the same goal (or subgoal) of a plan are the same as those in choosing actions orderings. Figure 2 shows two plans that look equally correct if the prospect of missing effects and preconditions is ignored, but one plan has more risks than the other and would be preferred by RP_w.

Table 1 reprints empirical results that measure how useful risk assessment can be for plan selection. For these experiments, we implemented a modified version of the Fast Forward planning system (Hoffmann & Nebel, 2001) that exploits augmented domain descriptions to find the risks in each generated plan. Then we generated large sets of candidate plans using D, measured their risks, and determined if they would succeed when executed using D_{true} (see Garland & Lesh (2002) for details). The first run measured the impact of risk assessment on action ordering decisions; the second run measured the impact on operator choice decisions. No LCW statements were added to D for these experiments.

Table 1 gives statistics showing how risks and success percentages are related for the two experiments. Within each row, there are four columns that show the likelihood of successfully executing a plan drawn at random from different subsets of the set of generated plans. For the column labelled t_k, the subset contains all plans whose number of risks are in the lowest kth percentile of distribution (e.g., t_{50} includes all plans with fewer risks than the median number of risks). The final five columns show general statistics about the distribution of risks in the set of generated plans.

The results show what a significant impact risk assessment can have. For both runs, generating two plans and preferring the one with fewer risks will, on average, increase the chance of success from roughly 75% to 80%. Generating more candidates continues to provide benefits, as selecting a plan from t_{10} increase the likelihood of success to over 85%.

3 Interleaved planning and execution

The above techniques are designed to choose from a set of plans to execute fully, given a complete model of the initial state. We now describe how to extend these techniques to make them suitable for continuous planning, i.e., a planner that interleaves planning and execution and solves more than one goal during its "life time". Of course there are many extremely challenging problems involved in creating a robust, continuous planning and execution system; here we focus exclusively on how a continuous planner can take advantage of the kind of risk-analysis we have outlined above.

As an illustrative example, suppose that a planning system had complete information about the state of the world at time t_0 but has since executed 1,000 actions $a_1, ..., a_{1000}$ in service of various goals. While it is the only actor in the world, it lacks a complete model of its actions. It now has a new goal G. For simplicity, assume that there are two actions α and β which achieve G. Each has one precondition, namely p_α and p_β.

The execution history $a_1, ..., a_{1000}$ can influence the choice of whether α or β is more likely to achieve G. Suppose that the most recently executed action with effect p_α and p_β was, respectively, a_{997} and a_{992}. It might seem that α is a wiser choice since fewer executed actions could have

clobbered p_α than p_β. However, there are many other factors to consider. The agent may know, for example, that actions $a_{993}, ..., a_{1000}$ do not effect p_β. Alternatively, suppose a_{992} has no preconditions but that a_{997}'s precondition was established by action a_{902}, and so if actions $a_{903}...a_{996}$ clobbered that precondition, then a_{997} may have not executed properly. (And, of course, a_{902}'s preconditions may, in turn, also be suspect.)

Additionally, sensor information should play a role in plan selection. For example, if a_{997}'s preconditions were sensed to be true at time 995, then the planner only needs to consider possible clobberings from a_{995} and a_{996}.

We now present techniques that perform the kind of reasoning needed to make such decisions. While the formulation in Garland & Lesh (2002) relied upon being given a complete model of the initial state, our current techniques assume initial state information is derived from sensor readings. Another key aspect of our approach is that the entire execution history does not need to be stored; all of the relevant information is encoded in a compact representation.

In our formulation, sensing actions are not explicitly modeled or planned for. Each atom x in the state is associated with a virtual sensor $\phi(x)$ (there may not be a 1-1 map of virtual to physical sensors — a single physical sensor may detect multiple atoms or input from multiple physical sensors may be "fused" to detect a single atom). $\phi(x)$ returns either true, false, or unknown. An unknown value indicates that no information about x is currently available. The sensed state of the world at any time is simply $\phi(x)$, for all atoms x.

The system receives information from its sensors after the execution of an action, and after the occurrence of an external event. An external event e can be modeled like an action and is detected by an associated virtual sensor $\phi(e)$. Thus, an external event in the execution history can be processed exactly like an attempted action. An external event can have modeled effects (e.g., rain makes a robot's gripper wet), can have associated LCW knowledge (e.g., rain does not make the robot's internal circuits wet), and can have unmodeled effects.

The agent maintains a mental state of the world $\mathcal{S}$ that differs from the sensed state of the world because of anticipated, but not observed, changes to the state of the world. For example, the mental state includes the effects of actions that the agent has executed. $\mathcal{S}$ also differs from the sensed state of the world since $\mathcal{S}$ keeps track of the risks associated with each literal x that is true (this point is explained more fully below).

In order to generalize our techniques, we identify a new type of risk:

- PRECUNKNOWN(a_i, x) :: a_i has a precondition whose value has never been sensed and is not an effect of any observed external event, executed action, or planned action a_j for $j < i$.

The function FINDPLANRISKS, shown in Figure 3, produces the set of risks of a plan $c = a_1, \ldots, a_n$, a goal g, the current mental state $\mathcal{S}$, and a set of locally closed world

```
FINDACTRISKS (a_i, S, L) ≡
  forall literals x in PRECONDITIONS(a_i)
    if UNKNOWNINSTATE(x, S)
      R ← R ∪ { PRECUNKNOWN(a_i, x) }
    else
      R ← R ∪ RISKSINSTATE(x, S)
  if CompletePreconditions(a_i) ∉ L
    R ← R ∪ { PRECOPEN(a_i) }
  return R

UPDATESTATE (a_i, S, L) ≡
  S ←COPYSTATE(S)
  R ←FINDACTRISKS(a_i, S, L)
  forall literals x in S
    if x ∈EFFECTS(a_i)
      SETTRUEINSTATE(x, S, R)
    else if DoesNotMakeFalse(a_i, x) ∉ L
      ADDRISKTOSTATE(x, S,POSSCLOB(a_i, x))
  return S

FINDPLANRISKS (<a_1, ..., a_n>, g, S, L) ≡
  for i = 1 to n
    S ←UPDATESTATE(a_i, S, L)
  RiskSet ← ∅
  forall literals x in g
    RiskSet ← RiskSet ∪ RISKSINSTATE(x, S)
  return RiskSet
```

Figure 3: Finding risks.

statements L.[2] For each action a_i in the plan, FINDPLANRISKS computes the set of risks R associated with executing a_i using FINDACTRISKS. The truth value of each precondition x of a_i is checked and if it is unknown, then a PRECUNKNOWN(a_i, x) is added to R; if x is true, then all of the risks associated with x in the current mental state $\mathcal{S}$ are added to r. Also, PRECOPEN(a_i) is added to R if a_i's preconditions are not completely modeled in L.

The system's initial mental model is simply the sensed state, with an empty risk set for each atom. Moving forward in time, the system updates the current mental model $\mathcal{S}$ after attempting an action a_i using the procedure UPDATESTATE. Basically, each effect of a_i becomes associated with risk set FINDACTRISKS$(a_i, \mathcal{S}, L)$. In addition, if there are literals x that might be clobbered by a_i then a POSSCLOB(a_i, x) is added to the risk set associated with x in $\mathcal{S}$. In addition to the changes to the risk sets made in UPDATESTATE, the risk set of any literal x is cleared whenever its corresponding sensor returns true or false.

FINDPLANRISKS is an improved version of the algorithm presented in (Garland & Lesh, 2002), which is inadequate for continuous plan selection. Since it makes one forward pass over the acts in the plan, and computes the union of the risk sets for the preconditions of each act, this algorithm requires $O(nmS)$ time, where n is the length of the plan, m is the number of atoms in D, and S is the size of the mental

[2] The pseudo-code has been stream-lined for pedagogical purposes; in general, it is necessary to check for hypothesized effects, false preconditions, and that the plan achieves g if D is complete.

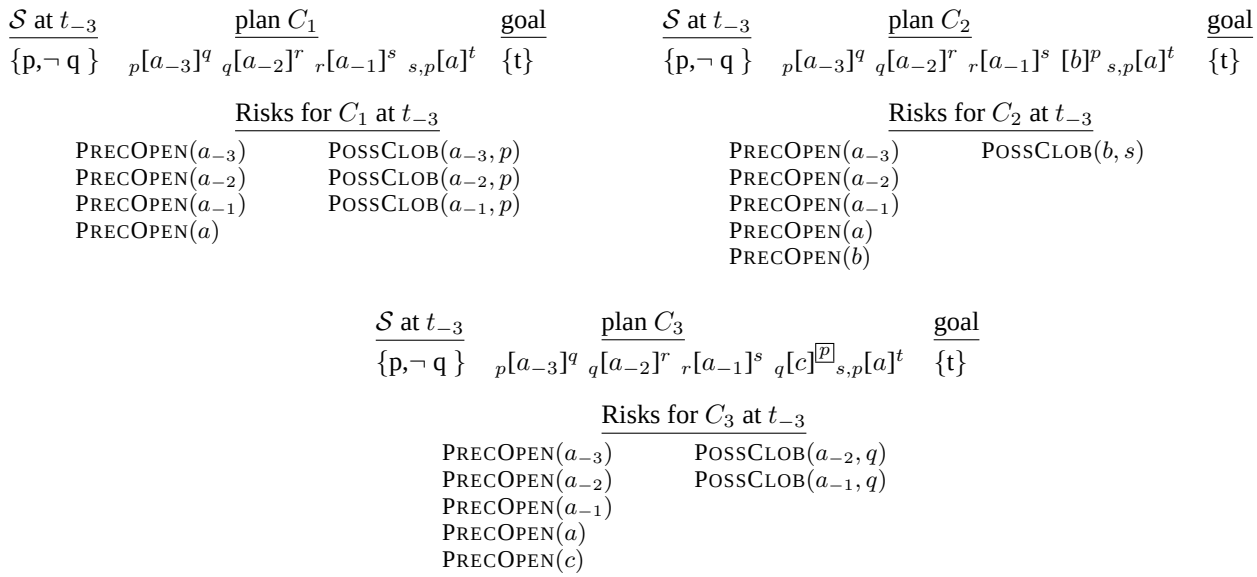

Figure 4: **Additional steps.** The extra step in C_2 (and C_3) re-establishes p, but introduces other risks.

state at time t. S is bounded by the number of different actions (including external events) that have occurred since the world was last sensed completely. In the worst case, S is bounded by the minimum of the number of possible instantiated actions or t.

This framework can account for the passage of time in a limited manner (there has been no need to implement this yet). This can be done by having a virtual sensor $\phi(e_t)$ that registers an "observed" external event e_t every t time units. e_t has no modeled effects, so our algorithm will identify different risk sets depending on exactly what LCW statements are given for e_t. One can use this mechanism to introduce a POSSCLOB risk every t time units for each literal that may become false without being sensed.

An area for future work is to reconcile the mental model of the state with the current observed state. This is a difficult inference problem, even in the absence of risk assessment. For example, imagine if an act a had an anticipated effect x that was sensed to be false immediately before attempting a and true immediately after a. It is tempting to conclude that all of a's known preconditions must have been true immediately before executing a, but this is only valid if one assumes that a has no effects when any precondition is false. When the system does have enough knowledge to infer that a successfully executed, then each risk in a's risk set is known to be spurious (i.e., it is not a possible source of execution failure), and can be safely removed from the risk set of *all* literals.

Our goal is to produce a system which considers all of these issues when choosing which action to execute. Ideally, as will be shown by the example in the next section, plan selection should be repeated after *any* observation since the risk sets will change. This does not necessitate re-planning, since some of the plans in the previous set of candidates will still be viable alternatives. If re-planning is not too computationally demanding, a reasonable heuristic would be to re-plan whenever an observation differs from the expected value.

4 Example

Imagine that it is imperative to execute action a soon, but an equally important consideration is that a must be successfully executed. The decision facing the system is whether or not to attempt a immediately, or to first execute a subplan designed to ensure that a's preconditions are true.

Figure 4 shows a specific example of this general situation in which the choice of which action to execute next depends on the past execution history and sensor readings. In this example, action a has two preconditions p and s. Sensor readings taken three time units ago, i.e., at time t_{-3}, indicated that p was true. Since then, actions a_{-3}, a_{-2}, and a_{-1} have been executed, which are known to establish a's other precondition, s, and are not known to clobber p. Figure 4 contains three plans (C_1, C_2, and C_3) for executing a. If risk analysis were ignored, plan C_1 would be preferred because a's preconditions are believed to be true. The problem with C_1, however, is that a_{-3}, a_{-2}, or a_{-1} might have clobbered a's precondition p. Plans C_2 and C_3 seek to re-establish p by alternative subplans, each of which is composed of a single action. C_2 relies on action b to re-establish p while C_3 uses action c. The advantage of using b is that it has no known preconditions, while c has precondition q that was established at t_{-2} and may have been clobbered by a_{-2} or a_{-1}. The disadvantage of using b is that it might have an unmodeled effect that clobbers a's other precondition, s, while action c's effects are completely known (this is indicated by the box around p) and thus could not clobber s.

Figure 4 lists the risks associated with the three plans that would have been identified at time t_{-3}, before executing any

of the actions. The risks associated with the plans in the current state of the world (i.e., after executing a_{-1}) depend upon what has been observed since t_{-3}.

Each of the three candidate plans can appear to be the best plan to execute depending on the sensed values for p and q. If p is sensed to be true in the current state, clearly C_1 should be preferred. In the absence of any sensor feedback after time t_{-3}, C_2 will be preferred by RP_w if POSSCLOB risks are weighted more than half of PRECOPEN risks. Finally, if p has not been sensed since t_{-3} and q is sensed to be true in the current state, C_3 appears to be the most likely plan to succeed because its precondition is known to be true and it cannot clobber s.

5 Discussion and related research

This work describes one component of a complete system for integrated planning and execution. Obviously, the system will need to be equipped with a planner that can reason efficiently in the face of incomplete state information. Also, the system should improve the action descriptions when possible.

Much previous work has addressed the problem of planning with incomplete state information and non-deterministic or conditional effects (e.g, Kushmerick, Hanks, & Weld (1995); Smith & Weld (1998)). This is similar to the problem of planning with incomplete action models in the sense that both problems are concerned with uncertainty about the effects of actions. One important difference, however, is that non-deterministic planning systems demand even more elaborate action descriptions than traditional planners. For example, the action descriptions are required to describe all the different possible sets of effects that an action might have. In contrast, our techniques can improve planning even without any additional information, and we provide a framework in which any additional effort to produce LCW statements can be factored into the planning process. Further, our techniques are designed to exploit various types of information, even statements about which fluents an action does *not* effect.

Additionally, the objective of work on non-deterministic planning is usually to generate plans that are guaranteed to succeed, or are guaranteed to succeed with some probability. As a result, even assessing a probabilistic or conditional plan to determine if it will succeed requires exponential computation win the length of the plan. In contrast, our methods simply prefer to execute plans with fewer risks. Further, our techniques are linear in the length of the plan and the size of the state, though we do require the planner to generate multiple plans to choose from.

Prior work has addressed the complementary problem of improving action models between planning episodes. One approach has been to develop knowledge acquisition systems that help domain experts convey their knowledge to a computer. For example, Tecuci *et al.* (1999) present techniques for producing hierarchical if-then task reduction rules by demonstration and discussion from a human expert. A second approach is to develop techniques for improving action descriptions based on the observed results of executing actions (Gil, 1994; Wang, 1995; Oates & Cohen, 1996).

Our techniques are complementary since our methods are designed to improve planning when there are incomplete action descriptions. However, a plan selection policy must address the classic exploration / exploitation tradeoff for a learning system that seeks to perform at the highest possible level over both the short run and the long run. One possible synergy between these two lines of research would be to develop techniques for automatically learning LCW or helping to elicit it from the domain experts.

Essentially, this work revisits the infamous *frame problem* (McCarthy & Hayes, 1969), which motivated the traditional completeness assumptions (Reiter, 1991). Without the completeness assumptions, it is necessary for the effects of an action description to include all facts whose truth value does *not* change as a result of executing it. The conceptual shift we make is to a willingness to execute plans with risk of failure if the action model is incomplete. Our methods simply prefer to execute plans with fewer risks.

6 Conclusion

This paper presented extensions to Garland & Lesh (2002) to support continuous plan evaluation. The primary contribution of this work is an improved algorithm that handles handles sensor feedback and external events and identifies an additional type of risk. As a result, the entire execution history does not need to be stored; all of the relevant information is encoded in the system's mental state of the world. Thus, these techniques form a time- and space- efficient plan selection component in a complete system for integrated planning and execution.

References

Babaian, T., and Schmolze, J. G. 2000. PSIPlan: Open World Planning with ψ-forms. In *Proc. 5th Int. Conf. on AI Planning Systems*, 292–307.

Garland, A., and Lesh, N. 2002. Plan Evaluation with Incomplete Action Descriptions. In *Proc. Eighteenth National Conference on Artificial Intelligence*, 461–467. http://www.merl.com/papers/TR2002-05.

Gil, Y. 1994. Learning by Experimentation: Incremental Refinement of Incomplete Planning Domains. In *Eleventh Intl Conf on Machine Learning*, 87–95.

Golden, K. 1998. Leap before you Look: Information Gathering in the PUCCINI planner. In *Proc. 4th Int. Conf. on AI Planning Systems*, 70–77.

Hoffmann, J., and Nebel, B. 2001. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research* 14:253–302.

Kushmerick, N.; Hanks, S.; and Weld, D. 1995. An Algorithm for Probabilistic Planning. *Artificial Intelligence* 76:239–286.

Levesque, H. J. 1996. What is planning in the presence of sensing. In *Proc. 13th Nat. Conf. AI*, 1139–1146.

McCarthy, J., and Hayes, P. J. 1969. Some philosophical problems from the standpoint of Artificial Intelligence. In

Meltzer, B., and Michie, D., eds., *Machine Intelligence 4*. Edinburgh University Press. 463–502.

Oates, T., and Cohen, P. 1996. Searching for planning operators with context-dependent and probabilistic effects. In *Proc. 13th Nat. Conf. AI*, 863–868.

Reiter, R. 1991. The Frame Problem in the Situation Calculus: A Simple Solution (Sometimes) and a Completeness Result for Goal Regression. In Lifschitz, V., ed., *Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy*. Academic Press. 359–380.

Smith, D., and Weld, D. 1998. Conformant Graphplan. In *Proc. 15th Nat. Conf. AI*, 889–896.

Tecuci, G.; Boicu, M.; Wright, K.; Lee, S.; Marcu, D.; and Bowman, M. 1999. An integrated shell and methodology for rapid development of knowledge-based agents. In *Proc. 16th Nat. Conf. AI*, 250–257.

Wang, X. 1995. Learning by observation and practice: an incremental approach for planning operator acquisition. In *Proc. 12th Int. Conf. on Machine Learning*, 549–557.