# Reachability Heuristics for Planning in Incomplete Domains

## Jared Robertson & Daniel Bryce

{jrobertson, daniel.bryce}@usu.edu

Utah State University

## Abstract

Engineering complete domain descriptions is often very costly because they are prone to errors of omission or commission. While many have studied knowledge acquisition, relatively few have studied the synthesis of low risk plans when actions may have missing or incorrect preconditions or effects. In this work, we focus upon omitted features (i.e., preconditions and effects) of actions, where the action features are in one of three groups: those that are i) known to be included, ii) known not to be included, and iii) not known to be either. Prior work has evaluated the correctness of *complete* plans synthesized with the incomplete domain theory, but no prior work has studied how a planner can lower the risk of failure by reasoning about its knowledge of the incompleteness. That is, by guiding search in terms of the potential risks of *partial* plans it should be possible to find lower risk plans.

We present and empirically evaluate a forward heuristic search planner called `FFRISKY`, which computes estimates of plan risk on planning graphs. Aside from being the first to address the problem, the unique aspect of the planner's heuristic is how it selectively propagates the risk to balance between risks incurred supporting propositions and generating risks that will effect propositions supported later. We compare `FFRISKY` with a control planner that uses the FF heuristic to measure plan length and ignore risk.

## Introduction

The knowledge engineering required to create complete and correct domain descriptions for planning problems is often very costly and difficult (Kambhampati, 2007). Machine learning techniques have been applied with some success (Wu, Yang, and Jiang, 2007), but still suffer from impoverished data and limitations of the algorithms (Kambhampati, 2007). In particular, we are motivated by applications in instruct-able computing (Mailler et al., 2009) where a domain expert teaches an intelligent system about a particular domain, but can often leave out whole procedures (plans) and aspects of action descriptions. In such cases, the alternative to making domains complete, is to plan around the incompleteness. That is, by assuming that enough knowledge of the domain exists to find a causally correct plan, we find low-risk plans that are most likely to succeed given the knowledge of how the domain is incomplete.

While prior work (Garland and Lesh, 2002) has categorized risks to a plan's correctness and described plan quality metrics in terms of the risks, no prior work has sought to deliberately synthesize low-risk plans. Specifically, the prior work of Garland and Lesh (2002) (henceforth abbreviated, GL) identifies four types of plan risks: possible open preconditions (due to incomplete knowledge of preconditions), possible clobbers (due to incomplete knowledge of delete effects), unlisted effects (due to incomplete knowledge of add effects), and false preconditions (due to the known domain model). GL develop an algorithm that steps backward through the plan to identify the critical risks – those risks that are a single source of failure for the plan. Critical risks prevent the plan from achieving vulnerable conditions. Vulnerable conditions are preconditions or goals where all sources of support share the same risks. The number of critical risks is an important measure of plan quality, that, unfortunately, no known planners seek to minimize.

We present a forward heuristic planner, called `FFRISKY`, that computes estimates of future plan risk on planning graphs. There are two interesting features of `FFRISKY` described in this work, its search node representation and heuristic. In both aspects, `FFRISKY` maintains information about the set of potential risks to achieving state propositions and the set of critical risks realized by the plan. `FFRISKY` carries possible risks to propositions forward so that if actions require the propositions as preconditions (thus making the risks to each proposition critical), it is possible to locally compute the critical risks without examining the entire plan (as one might using the algorithm of GL). We describe the propagation of possible risks and the calculation of critical risks both in the search space and within relaxed planning graphs. The number of critical risks in the plan prefix is the quality of a partial plan, and the number realized in achieving the goals in the planning

graph is the heuristic.

We evaluate FFRISKY in several instances of modified IPC domains against a control planner that uses a similar search algorithm, but optimizes plan length with the FF heuristic (Hoffmann and Nebel, 2001). The results indicate that FFRISKY can find much lower risk plans, at the expense of plan length (reducing risk often involves using multiple actions to achieve the same subgoal). Moreover, FFRISKY must often generate more search nodes than the control planner, but incurs only a small additional cost in computing its heuristic. In the following, we provide background on the representation of the planning problems studied, definitions of plan risks, our search formulation, heuristic computation, empirical evaluation, related work, and conclusion.

## Background

A complete planning domain $D_{true}$ defines the tuple $(P, A, I, G)$, where $P$ is a set of propositions, $A$ is a set of complete action descriptions, $I$ is the initial state, and $G$ is a set of goals. An incomplete planning domain $D$, corresponding to $D_{true}$, defines the tuple $(P, \tilde{A}, I, G)$, where $\tilde{A}$ is a set of incomplete action descriptions. (We use "$\sim$" throughout to denote incomplete aspects of the incomplete domain theory).

**Action Representation**: Complete action descriptions follow the standard STRIPS representation (Fikes and Nilsson, 1971), where each action $a \in A$ defines the tuple (pre($a$), add($a$), del($a$)), where the precondition, add effects, and delete effects are sets of propositions. Incomplete action descriptions extend the representation of complete actions to account for possible preconditions and effects, such that each incomplete action $\tilde{a} \in \tilde{A}$ defines the tuple (pre($\tilde{a}$), $\widetilde{\text{pre}}(\tilde{a})$, add($\tilde{a}$), $\widetilde{\text{add}}(\tilde{a})$, del($\tilde{a}$), $\widetilde{\text{del}}(\tilde{a})$), where pre($\tilde{a}$) is the set of known preconditions and $\widetilde{\text{pre}}(\tilde{a})$ is the set of possible preconditions, similarly for known and possible add and delete effects. We differ syntactically from GL with our action descriptions because we choose to represent the true and possible preconditions, where GL represents the true and impossible preconditions (via local closed world statements). The two representations are equivalent because the impossible preconditions are all other propositions not mentioned in the true or possible preconditions, and similarly for effects.

**Plan Semantics**: A complete action is applicable to a state, appl($s_i, a_i$) if pre($a$) $\subseteq s_i$. Applying a complete action $a_i$ to a state $s_i$ results in a successor state $s_{i+1} = \text{succ}(s_i, a_i) = s_i \backslash \text{del}(a_i) \cup \text{add}(a_i)$. A feasible plan is a sequence of complete actions $(a_0, ..., a_{n-1})$ that achieves the goals $G$, that is $s_0 = I$, $G \subseteq s_n$, and each action is applicable, appl($s_i, a_i$).

Similarly, an incomplete action is applicable to a state appl($s_i, \tilde{a}_i$) if pre($\tilde{a}_i$) $\subseteq s_i$ (i.e., we do not require that possible preconditions are satisfied because unsatisfied preconditions are incorporated into the measure of plan risk). Applying an incomplete action $\tilde{a}_i$ to a state $s_i$ results in a successor state $s_{i+1} = \text{succ}(s_i, \tilde{a}_i) = s_i \backslash \text{del}(\tilde{a}_i) \cup \text{add}(\tilde{a}) \cup \widetilde{\text{add}}(\tilde{a})$. Our semantics for incomplete actions are *optimistic* because we do not require that possible preconditions are satisfied and we include the possible add effects when computing successor states. A feasible plan with incomplete actions, like a plan with complete actions, must achieve the goals $G$, per above.

While a plan with incomplete actions must be causally correct (wrt. the optimistic semantics), such that each action is applicable in the plan and the goals are satisfied, its quality is determined by risks to its correctness (wrt. the complete semantics). As we define in the following, a plan's quality is the number of risks that are critical to its success or failure.

## Plan Risks

A *risk* is a potential source of execution failure due to incompleteness of the action model. A *critical risk* is not guaranteed to cause plan failure, but it does guarantees the possibility of failure (Garland and Lesh, 2002). We denote the set of all plan risks by $R$. The risks are:

- 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($a_i, x$) :: action $a_i$ might have an unlisted precondition $x$ that will not be true when $a_i$ is executed in $a_0, \ldots, a_n$.

- UnlistedEffect($a_i, x$) :: $a_i$ might have effect $x$ that is not listed in $D$.

We do not track false precondition risks, as described by GL, because they imply that either the plan is not causally correct wrt. $D$, or that an unlisted effect is used to support the precondition. If an unlisted effect is used to support a precondition, then introducing a false precondition risk counts the risk twice.

We also differ from GL by counting a PrecOpen risk for each possibly open precondition of an action, instead of a single PrecOpen risk for the action. This choice is influenced by two concerns: having more open preconditions should be undesirable, and because we represent the possible preconditions rather than the impossible preconditions, we can easily calculate which possible preconditions are not satisfied.

## Forward State Space Risk Maintenance

In order to search for low-risk plans, we require a search formulation that supports efficient computation of the

number of critical risks to the plan. While it is straight-forward to use a plan space search and use the algorithms suggested by GL to compute risks (modulo our differing definitions), we develop a state space search formulation that supports additive computation of plan quality. That is, our state space search computes the plan risks in terms of the most recent plan step.

**Search Nodes**: Each search node $q$ is the tuple $(s, r_s, c_q)$, where $s$ is the state, $r$ is a function mapping state propositions to sets of possible risks, and $c_q$ is a set of critical risks realized by the paths leading to the node. The risk function $r_s : P \rightarrow 2^R$ is defined over all propositions $p \in s$, mapping each to a subset of the risks $R$ realized by the action sequence leading to the search node. The set of critical risks $c_q \subseteq R$ is the set of risks that will cause plan failure, and the cardinality of $c$ is the quality of all plans reaching the node (i.e., the g-value).

**Node Expansion**: Search nodes are computed as follows. The root node, where $s = I$, defines $r_s(p) = \{\}$ for each $p \in s$ and $c = \{\}$. In other words, the root node does not associate and possible or critical risks with the initial state.

Given a node $q = (s, r_s, c_q)$ and applicable action $\tilde{a}$, the successor node is defined $q' = (s', r_{s'}, c_{q'})$, where $s' = \text{succ}(s, \tilde{a})$.

The set of possible risks associated with each proposition in the successor state represent the risks contributed by all sources of support to the proposition. In sequential planning, as herein, there are two sources of support: the persistence of the proposition from the prior state and the add effects of the applied action. Risks to a proposition can only become critical if they are contributed by all sources of support. Therefore, the possible risks for each proposition are obtained by the following expression, where we take the intersection of the persisted risks, and the risks contributed by the action, $r_{s'}(p) =$

$$r_s(p) \cap \left( \bigcup_{p' \in \text{pre}(\tilde{a})} r_s(p') \cup \bigcup_{p' \in \widetilde{\text{pre}}(\tilde{a}):p' \in s} r_s(p') \cup r_{s,\tilde{a}}(p) \right).$$

The risks contributed by the action include all risks to the action's precondition and all risks generated by the action $r_{s,\tilde{a}}(p)$ that are relevant to the proposition. The risks contributed by the action $r_{s,\tilde{a}}(p)$ include a $\texttt{PrecOpen}(\tilde{a}, p')$ risk if there is a possible precondition in $p' \in \widetilde{\text{pre}}(\tilde{a})$ that is not supported by $s$, an $\texttt{UnlistedEffect}(a, p)$ risk if $p$ is a possible add effect; or a $\texttt{PossClob}(\tilde{a}, p)$ risk if $p$ is a possible delete effect. The critical risks $c_{q'}$ realized by the plan prefixes reaching $q'$ is defined as the prior critical risks and all possible risks to the action's preconditions,

$$c_{q'} = c_q \cup \bigcup_{p \in \text{pre}(\tilde{a})} r_s(p).$$

The g-value of the search node is defined as the number of critical risks of the plan(s) reaching the node,

$$g(q') = g(q) + |c_{q'} \setminus c_q| = |c_{q'}|.$$

## Propagating Risk in Planning Graphs

Similar to propagating risks that effect vulnerable conditions within the search space, we can compute a heuristic measure of the critical risks effecting goal achievement within relaxed planning problems. The planning relaxation provided by planning graphs lets us propagate risks and lower bound the actual number of critical risks.

**Planning Graph Heuristics**: A relaxed planning graph $RPG$ is a layered graph of sets of vertices $(\mathcal{P}_0, \mathcal{A}_0, ..., \mathcal{A}_{k-1}, \mathcal{P}_k)$. The planning graph built w.r.t. a state $s$ defines $\mathcal{P}_0 = s$, $\mathcal{A}_t = \{a | \text{pre}(a) \subseteq \mathcal{P}_t\}$, and $\mathcal{P}_{t+1} = \{p | a \in \mathcal{A}_t, p \in \text{add}(a)\}$. A simple heuristic, $h^{\max}$ for the number of actions to achieve the goal $G$ from $s$ is equivalent to the minimum level $t$ where the goals are reached, $h^{\max} = \min_{t:G \subseteq \mathcal{P}_t} t$ (Bonet and Geffner, 1999). The FF heuristic $h^{ff}$ (Hoffmann and Nebel, 2001) solves this relaxed planning problem by choosing actions from $\mathcal{A}_t$ to support the goals in $\mathcal{P}_{t+1}$, and recursively for each action's preconditions, in order to count the number of chosen actions as the heuristic.

When planning with incomplete domains, we would like to minimize the number of critical risks. A heuristic should measure the number of new critical risks encountered while trying to achieve the goal. We can propagate measures of risk over the relaxed planning graph to determine the risks to achieving the goals at each level $t$. However, like our search formulation, we need to track both the possible risks to, and the (realized) critical risks in, achieving propositions. As in the search space, those risks to achieving a proposition become critical (i.e., are realized) when they are used to support an action. However, because the planning graph represents all reachable propositions, we associate a set of critical risks with each proposition, where in the state space we associated a set of critical risks with each state. The critical risks associated with goal propositions are used to compute the heuristic. We propagate possible and realized critical risks differently because the former set is used to compute the latter set. The following intuitions dictate the propagation semantics for possible and realized risks:

- Possible risks to propositions in the evaluated state become critical when used to support actions in the planning graph.

- All possible risks to an action's preconditions become critical when the action appears. Therefore, the possible and realized risks to executing an action are identical.

- The critical risks to executing an action include those realized via the action's preconditions (which were previously only possible) and those realized by actions supporting the preconditions.

- If an action is used to support a proposition, then the possible risks to the proposition include an unlisted

effect risk (if supported by a possible effect) and the possible risks to the action, the critical risks to the proposition include the risks that were critical to the action.

A relaxed planning graph with propagated risks $\widetilde{RPG}$ is a layered graph of sets of vertices $(\tilde{\mathcal{P}}_0, \tilde{\mathcal{A}}_0, ..., \tilde{\mathcal{A}}_{k-1}, \tilde{\mathcal{P}}_k)$, a possible risk function $r_{\widetilde{RPG}}$ : $\{\tilde{A}, \tilde{P}\} \times \mathbb{N} \to R$, that maps each vertex to a set of possible risks to achieving it, and a critical risk function $c_{\widetilde{RPG}} : \{\tilde{A}, P\} \times \mathbb{N} \to R$ denoting risks realized in achieving a proposition or the preconditions of an action.

**Initial Proposition Layer**: The $\widetilde{RPG}$ built for a state $s$ defines the layers and risks for each vertex, such that the initial proposition layer contains all propositions in the state $\tilde{\mathcal{P}}_0 = s$, the possible risks associated with each proposition are those of the state $r_{\widetilde{RPG}}(p, 0) = r_s(p)$, and there are no realized critical risks for any proposition $c_{\widetilde{RPG}}(p, 0) = \{\}$.

**Action Layer**: Action layers contain those actions whose preconditions are satisfied in the proposition layer $\tilde{\mathcal{A}}_t = \{\tilde{a} \in \tilde{A} | \text{pre}(\tilde{a}) \subseteq \tilde{\mathcal{P}}_t\}$. The possible risks associated with each action include any open precondition risks and those risks to the action's preconditions,

$$r_{\widetilde{RPG}}(\tilde{a}, t) = \{\texttt{OpenPrec}(\tilde{a}, p) | p \in \widetilde{\text{pre}}(\tilde{a}), p \notin \tilde{\mathcal{P}}_t\} \cup \bigcup_{p \in \text{pre}(\tilde{a})} r_{\widetilde{RPG}}(p, t).$$

The critical risks to executing the action include all possible risks to the action and all critical risks realized by supporting the preconditions,

$$c_{\widetilde{RPG}}(\tilde{a}, t) = r_{\widetilde{RPG}}(\tilde{a}, t) \cup \bigcup_{p \in \text{pre}(\tilde{a})} c_{\widetilde{RPG}}(p, t).$$

**Proposition Layer**: The subsequent proposition layer contains all propositions appearing in the true or possible add effects of an action (including noop actions), $\tilde{\mathcal{P}}_{t+1} = \{p | p \in \text{add}(\tilde{a}) \cup \widetilde{\text{add}}(\tilde{a}), \tilde{a} \in \tilde{\mathcal{A}}_t\}$. The possible risks to achieving a proposition $p$ at $t + 1$ are determined by the set of *chosen* supporting actions $\tilde{S}_t(p)$ (described below), which may be any non-empty subset of the actions in level $t$ that support $p$. Given the set of supporters $\tilde{S}_t(p)$, the possible risks to achieving the proposition include risks from two types of supporters: a true supporter ($p \in \text{add}(\tilde{a})$) and a possible supporter ($p \in \widetilde{\text{add}}(\tilde{a})$). The true supporters contribute possible risks to executing the supporting action and the possible supporters contribute the same risks and any risks related to having possible effects (i.e., unlisted effects),

$$r_{\widetilde{RPG}}(p, t+1) =$$

$$\bigcap_{\tilde{a} \in \tilde{S}_t(p) : p \in \text{add}(\tilde{a})} r_{\widetilde{RPG}}(\tilde{a}, t) \cap$$

$$\left( \bigcap_{\tilde{a} \in \tilde{S}_t(p) : p \in \widetilde{\text{add}}(\tilde{a})} r_{\widetilde{RPG}}(\tilde{a}, t) \cup \{\texttt{UnlistedEffect}(\tilde{a}, p)\} \right).$$

The critical risks realized by supporting a proposition include all critical risks to to executing the supporting actions,

$$c_{\widetilde{RPG}}(p, t+1) = \bigcup_{\tilde{a} \in \tilde{S}_t(p)} c_{\widetilde{RPG}}(\tilde{a}, t).$$

The set or supporting actions $\tilde{S}_t(p)$ can be defined multiple ways, each having an impact on which possible and critical risks are propagated. As in the search space, using multiple actions to support a proposition may remove possible risks to achieving the proposition, but each action may introduce critical risks. Clearly, there exist trade-offs in choosing supporters for propositions, both in the search space and the relaxed planning graph.

**Choosing Supporters**: While only a single action may be required to causally support a proposition, using multiple supporters can mitigate the *propagation of possible risks* by ensuring that not all sources of support are subject to the same risks. However, including more actions can *immediately realize more critical risks.* That is, by including more supporters, the set $r_{\widetilde{RPG}}(p, t+1)$ may shrink, but the set $c_{\widetilde{RPG}}(p, t+1)$ may grow. The measure of interest $|c_{\widetilde{RPG}}(p, t+1)|$ indicates the *cost* in terms of critical risks to achieve a proposition, but the set of possible risks $r_{\widetilde{RPG}}(p, t+1)$ translates to potential critical risks for propositions appearing later in the planning graph. Thus, the tradeoff to selecting supporters is between incurring immediate cost and incurring future cost. This tradeoff is the crux of the problem in planning with risks: actions selected to remove risks often introduce new risks, but it is not clear whether introduced risks will become realized until the plan is complete.

We use the following greedy approach to selecting supporters for a proposition. We first choose the supporting action that introduces the fewest number of critical risks, and break ties by minimizing the number of possible risks. To the set of chosen supporters, we add the action whose intersection with the set of possible risks is smallest, breaking ties with the number of new critical risks introduced. In this fashion, we start by minimizing the number of immediate critical risks and add actions until the number of future possible risks will not decrease further.

**Heuristic Computation**: The $\widetilde{RPG}$ reaches a fixpoint, when the critical risk function does not change, $c_{\widetilde{RPG}}(p, t) = c_{\widetilde{RPG}}(p, t+1)$. The heuristic $h^{\widetilde{level}}$ mea-

| Prob | FFRISKY T/N/L/Q | Control T/N/L/Q |
|---|---|---|
| DL1 | 0.57 / 11 / 9 / 0 | 0.18 / 12 / 9 / 0 |
| DL2 | 33.65 / 999 / 21 / 1 | 0.47 / 51 / 20 / 2 |
| DL3 | 204.60 / 6001 / 18 / 1 | 5.60 / 574 / 16 / 4 |
| DL4 | 1416.59 / 27733 / 11 / 1 | 5.41 / 942 / 18 / 2 |
| DL5 | No Sol. | 20.33 / 1071 / 16 / 4 |
| R1 | 13.45 / 543 / 18 / 0 | 0.34 / 34 / 12 / 3 |
| R2 | 6.55 / 833 / 12 / 2 | 2.03 / 311 / 12 / 2 |
| R3 | 15.63 / 1099 / 9 / 0 | 2.68 / 208 / 10 / 1 |
| R4 | 19.90 / 76 / 12 / 2 | 3.08 / 345 / 10 / 3 |
| R5 | 989.55 / 4627 / 25 / 1 | 23.44 / 630 / 22 / 3 |

Table 1: Results For FFRISKY and the control planner.

sures the realized critical risks to achieving the goals at the fixed point level $t$, such that

$$h^{\widetilde{level}} = |\bigcup_{p \in G} c_{\widetilde{RPG}}(p, t)|.$$

## Empirical Evaluation

The empirical evaluation is divided into three sections that describe the domains used for the experiments, the test setup used, and a presentation and discussion of the results. We compare Frisksy with a control planner that uses A* search and the FF heuristic to search in terms of plan length. The questions that we would like to answer include:

- Does FFRISKY find lower risk plans than the control planner?

- Does the FFRISKY heuristic scale to find non-trivial plans?

- Does reducing risk imply that plans have greater length?

## Domains

We modified two well-known IPC domains: Rovers and DriverLog. Rovers involves a rover that must traverse between several locations, take samples, take pictures, and communicate data. In the Rovers modifications most actions have a possible delete effect on calibrated cameras and communicating data back to the lander has a possible precondition that the lander is visible. DriverLog involves several trucks, drivers, and packages where drivers board and drive trucks and packages are loaded and unloaded. The DriverLog modifications involve possible effects that the driver breaks down when driving and the driver can fix the car with a possible fix effect. In DriverLog, there is also a possible precondition to boarding the truck that it is empty, and disembarking has a possible effect that it is empty – this means that the truck can have more than one driver, but it is possible in the true domain that there can be only one driver per truck.[1]

## Test Setup

The tests were run on a machine running Linux with a 3 Ghz Xeon processor and a memory limit of 1GB. All code was written in Java and run on the 1.5 JVM. Both FFRISKY and the control planner share the same A* implementation, but use different search node representations. Both planners also use the same FF heuristic implementation, but FFRISKY uses the FF heuristic and the partial plan length to break ties between nodes that have the same f-value (when computing risks).

The planners are compared by four measures: total run time in seconds (T), the number of nodes expanded (N), the solution plan length (L), and the solution plan quality as the number of critical risks (Q).

## Results

The results depicted in Table 1 compare FFRISKY to the control planner on several problems (listed in each row). The data presented for each planner are the measures mentioned above. Instances in the DriverLog domain are prefixed by "DL" and instances in the Rovers domain are prefixed by "R".

**Rovers**: The Rovers domain involves relatively fewer risks than the DriverLog domain, but we continue to see improvement in the number of critical risks with FFRISKY. In all cases, FFRISKY finds plans with fewer risks. The heuristic used by FFRISKY appears to be fairly weak because of the high number of additional expanded nodes, and the heuristic considerably raises the per node cost. Yet, there are cases, like R4, where FFRISKY finds a plan in fewer nodes. It does appear that reducing risk increases plan cost, except in R3, where FFRISKY dominates in all but search time. Generally, FFRISKY does not require many additional actions to improve risk.

**DriverLog**: The DriverLog domain exhibits trends that are similar to Rovers. FFRISKY requires more search nodes, but always finds better risk plans in sometimes fewer plan steps. It appears that FFRISKY has a weaker heuristic in this domain because it requires a higher proportion of search nodes over the control planner than in Rovers and fails to solve DL5. The additional nodes is also likely to the plans admitting more risks.

**Summary**: While our sample size is relatively small, FFRISKY does appear to reduce risk over the control planner and can scale reasonably well (without other search enhancements, such as those used in FF and other planners). Interestingly, there are cases where FFRISKY finds plans with fewer risks and a lower plan length than the control planner, but these seem to

---

[1]The domains and instances are available at: http://www.cs.usu.edu/danbryce/supplementary/IncDoms.tgz.

be rare. We anticipate better scalability by using relaxed plans extracted from the risk propagated planning graph, but are encouraged by these preliminary results.

## Related Work

Planning with risks is noticeably similar to planning with incomplete information (Bonet and Geffner, 2000), where the action descriptions instead of the state are incomplete. The action incompleteness could be modeled as state incompleteness and our plans might be conformant. However, conformant planning techniques are not readily applicable because plans with risks will not be able to guarantee goal satisfaction, rather, only partial satisfaction. This is because plans with risks are weak (Bertoli et al., 2001), and there is generally no preference ordering over weak plans (i.e., there is no meaningful degree of weakness). Moreover, translating the problem to planning under uncertainty may decrease practical or theoretical performance, a topic that we are investigating.

Our investigation is an instantiation of model-lite planning, proposed by Kambhampati (2007), that focusses on incomplete domain theories. As pointed out by Kambhampati (2007), constraint-based hierarchical task networks are another approach to avoiding the specification of all preconditions and effects by designing methods and constraints that lead to the same plans that would be realized by the underlying, implicit causal links.

As previously stated, this work is a natural extension of the model for evaluating plans in incomplete domains (Garland and Lesh, 2002). Our methods for computing risks are slightly different in that we compute risks in the forward direction, are more specific about which open precondition risks occur (basing them on preconditions and not just the action itself), and drop the risks for false preconditions (because we require plans are causally correct in the incomplete domain). In addition to calculating risks of partial plans, we have also presented a relaxed planning heuristic that propagates risks on planning graphs.

## Conclusion

We have presented the first work to address planning in incomplete domains as heuristic search to minimize critical plan risks. Our planner, FFRISKY, i) performs forward search while maintaining sets of possible risks to state propositions and a set of realized critical risks, and ii) estimates the future critical risks incurred by propagating risks on planning graphs. We have shown that, compared to a planner that essentially ignores aspects of the incomplete domain, FFRISKY is able to scale similarly but find much better quality plans.

Future work on this topic will focus on additional heuristics for estimating risk that incorporate possible clobberer risks and other negative interactions. One direction for extending the heuristics will use interaction propagation to measure the cost added by possible or real mutexes (Bryce and Smith, 2006). We also intend to compare against the approach mentioned above that translates the incomplete action descriptions into incomplete states and uses a conformant planner. Directions for extending our model of incompleteness include adding probabilistic measures of various action features existing in the true domain and how incompleteness can be integrated with uncertain/stochastic action effects.

## References

Bertoli, P.; Cimatti, A.; Roveri, M.; and Traverso, P. 2001. Planning in nondeterministic domains under partial observability via symbolic model checking. In *Proceedings of IJCAI'01*.

Bonet, B., and Geffner, H. 1999. Planning as heuristic search: New results. In *Proceedings of ECP'99*.

Bonet, B., and Geffner, H. 2000. Planning with incomplete information as heuristic search in belief space. In *Proceedings of AIPS'00*.

Bryce, D., and Smith, D. E. 2006. Using interaction to compute better probability estimates in plan graphs. In *ICAPS 2006 Workshop on Planning Under Uncertainty and Execution Control for Autonomous Systems*.

Fikes, R., and Nilsson, N. 1971. STRIPS: A new approach to the application of theorem proving to problem solving. In *Proceedings of Second International Conference on Artificial Intelligence*, 608–620.

Garland, A., and Lesh, N. 2002. Plan evaluation with incomplete action descriptions. In *AAAI*.

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

Kambhampati, S. 2007. Model-lite planning for the web age masses: The challenges of planning with incomplete and evolving domain theories. In *Proceedings of AAAI'07*.

Mailler, R.; Bryce, D.; Shen, J.; and Orielly, C. 2009. Mable: A framework for natural instruction. In *Proceedings of AAMAS'09*.

Wu, K.; Yang, Q.; and Jiang, Y. 2007. Arms: an automatic knowledge engineering tool for learning action models for ai planning. *Knowl. Eng. Rev.* 22(2):135–152.