Non-Deterministic Planning With Conditional Effects

Christian Muise and **Sheila A. McIlraith** and **Vaishak Belle**
Department of Computer Science
University of Toronto, Toronto, Canada.
{cjmuise,sheila,vaishak}@cs.toronto.edu

Abstract

Recent advances in fully observable non-deterministic (FOND) planning have enabled new techniques for various applications, such as behaviour composition, among others. One key limitation of modern FOND planners is their lack of native support for conditional effects. In this paper we describe an extension to PRP, the current state of the art in FOND planning, that supports the generation of policies for domains with conditional effects and non-determinism. We present core modifications to the PRP planner for this enhanced functionality without sacrificing soundness and completeness. Additionally, we demonstrate the planner's capabilities on a variety of benchmarks that include actions with both conditional effects and non-deterministic outcomes. The resulting planner opens the door to models of greater expressivity, and does so without affecting PRP's efficiency.

1 Introduction

While classical planning assumes deterministic actions, many real-world planning problems are better suited to a model with non-deterministic action outcomes. Fully observable non-deterministic (FOND) planning problems require the planner to consider every possible action outcome, often guaranteeing achievement of the goal under an assumption of fairness. Recent advances have improved the efficiency of FOND planning (Kuter et al. 2008; Mattmüller et al. 2010; Fu et al. 2011; Muise, McIlraith, and Beck 2012), opening the door to new FOND-based techniques for a variety of applications, including behaviour composition (Ramírez, Yadav, and Sardiña 2013) and the synthesis of controllers satisfying temporally extended specifications (Patrizi, Lipovetzky, and Geffner 2013).

The PRP planner (or simply PRP) represents the current state of the art in FOND planning (Muise, McIlraith, and Beck 2012). Unfortunately, as with other FOND planners, PRP lacks native support for actions with conditional effects. In this paper, we present an extension to PRP that supports the generation of policies for domains with conditional effects and non-determinism. The resulting planner is capable of solving this richer family of planning problems, without compromising PRP's efficiency. This advance is significant, not only for FOND planning but for emerging applications that exploit FOND in their computational core.

A key aspect of PRP's success is that it focuses only on what is *relevant* for a weak plan to succeed. The relevant aspect of state is represented as a so-called *partial state* – equivalent to a conjunction of literals – computed via regression rewriting (Waldinger 1977; Reiter 2001). The challenge with incorporating conditional effects is that such a goal regression is considerably more involved. Essentially, the result of regression is no longer a partial state, but possibly an arbitrary formula – a set of partial states. Our main insight in this paper is to leverage the context in which the relevance information is being used to simplify the computation. In particular, we can use the state from which we are exploring the plan space to simplify the result of regression back to a single partial state. Indeed, this operation, which we term *pre-image filtering*, has a number of interesting properties, the most significant of which is that we do not have to introduce new computational machinery to handle arbitrary formulas resulting from regression. This not only maintains the soundness and completeness of PRP on a substantially richer set of problems, but also supports all of the techniques which make PRP an efficient FOND planner.

To evaluate the modifications we make to PRP, we investigate the planner's performance on four domains inspired by existing benchmarks. We find that the modifications made to PRP enable it to solve a wider class of problems than previously possible, without suffering from a deterioration in performance. To the best of our knowledge, no existing FOND planner is capable of approaching these results.

2 Background and Notation

In this section we review both our modelling language and the essentials of PRP (Muise, McIlraith, and Beck 2012; Muise 2014).

SAS$^+$ FOND Planning: As input, we assume we are given a non-deterministic planning problem as a PDDL file with "oneof" clauses in the action effects. We convert the domains to a non-deterministic SAS$^+$ formalism using a modified version of the PDDL-to-SAS$^+$ translation algorithm due to Helmert (2009).

We adopt the notation of (Muise, McIlraith, and Beck 2012) for non-deterministic SAS$^+$ planning problems, and

extend it to include conditional effects. A SAS$^+$ *fully observable non-deterministic* (FOND) planning task is a tuple $\Pi = \langle \mathcal{V}, s_0, s_*, \mathcal{A} \rangle$. $\mathcal{V}$ is a finite set of variables v, each having the finite domain D_v. D_v^+ is the *extended* domain of v that includes the value $\perp$, representing the undefined status of v. A *partial state* is a function s that maps variables to values, *i.e.* $s : \mathcal{V} \rightarrow D_v^+$. If $s(v) \neq \perp$ then v is *defined* for s, and if every variable $v \in \mathcal{V}$ is defined for s then s is a *complete state*. The initial state s_0 is a complete state, and the goal state s_* is a partial state. A partial state s *entails* another partial state s', denoted as $s \models s'$, iff $s(v) = s'(v)$ whenever v is defined for s'. Two partial states s and s' are *consistent* with one another, denoted $s \approx s'$, iff for all $v \in \mathcal{V}$, $s(v) = s'(v)$ or $s(v) = \perp$ or $s'(v) = \perp$. The *updated* partial state obtained from *applying* partial state s' to partial state s, denoted as $s \oplus s'$, is the partial state s'' where $s''(v) = s'(v)$ if v is defined for s', and $s''(v) = s(v)$ otherwise. Note that s and s' need not be consistent for the operator $\oplus$ to be applied. In what follows, it will be convenient to speak of the variables that are defined in a partial state s, which we denote using $vars(s)$.

Each action $a \in \mathcal{A}$ is of the form $\langle pre_a, \mathit{Eff}_1, \ldots, \mathit{Eff}_n \rangle$, where pre_a is a partial state, and Eff_i is an *effect* which is a set of the form $\{\langle cond_1, v_1, d_1 \rangle, \ldots, \langle cond_k, v_k, d_k \rangle\}$, where $cond_i$ is a partial state called an *effect condition*, $v_i \in \mathcal{V}$, and $d_i \in D_{v_i}$. A *fully observable deterministic* domain, then, is simply the case where $n = 1$ for all actions (with arbitrary k). In FOND domains, action outcomes are outside the control of the agent, and any of Eff_i can occur. The result of a particular effect *Eff* at partial state s is defined as:

$$\mathit{Result}(s, \mathit{Eff}) = \{v = d \mid \langle cond, v, d \rangle \in \mathit{Eff}, s \models cond\}$$

We say an action a is *applicable* in state s iff $s \models pre_a$ and a is *possibly applicable* in s iff $s \approx pre_a$. When a is (possibly) applicable in s, and for every $\langle cond, v, d \rangle \in \mathit{Eff}$, for which either $s \models cond$ or $s \not\approx cond$, we define the *progression* of s, wrt an action a and effect *Eff*, denoted $\mathit{Prog}(s, a, \mathit{Eff})$, as the updated state $s \oplus \mathit{Result}(s, \mathit{Eff})$. When $s \approx cond$ and $s \not\models cond$ for some $cond$, $\mathit{Prog}(s, a, \mathit{Eff})$ is undefined.

Unlike classical planning, where a solution is a sequence of actions, a solution to a FOND planning task is a *policy* that maps a state to an appropriate action such that the agent eventually reaches the goal. A *closed* policy is one that returns an action for every non-goal state reached by following the policy. We say that a state s is *reachable* by a policy if by following the policy, the agent may reach state s.

There are three types of plans for a FOND problem (Cimatti et al. 2003): *weak*, *strong*, and *strong cyclic*. Intuitively, a *weak plan* corresponds to an "optimistic plan" that reaches the goal under at least one selection of action outcomes (i.e., it will have some chance of achieving the goal). A *strong plan* corresponds to a "safe plan" and is a closed policy that achieves the goal in a finite number of steps while never visiting the same state twice. Often, however, weak plans are not acceptable and strong plans do not exist. As a viable alternative, a *strong cyclic plan* is a closed policy with the property that every reachable state will eventually reach the goal using the policy under an assumption of "fairness". The notion of fairness used here stipulates that if an action is executed infinitely often, then every non-deterministic effect of that action occurs infinitely often. In this work, we are interested in computing strong cyclic plans.

The approach that PRP uses to compute a strong cyclic plan is to solve a determinization of the FOND planning problem repeatedly with various initial states and goals. A *determinization* of a SAS$^+$ FOND planning task $\Pi = \langle \mathcal{V}, s_0, s_*, \mathcal{A} \rangle$ is a planning task $\Pi = \langle \mathcal{V}, s_0, s_*, \mathcal{A}' \rangle$ where the actions $\mathcal{A}$ are replaced by their corresponding deterministic actions $\mathcal{A}'$. Here, we exclusively use the *all-outcomes* determinization which creates $\mathcal{A}'$ by creating a new action for every non-deterministic effect of an action in $\mathcal{A}$.

3 PRP with Conditional Effects

The contribution of this paper is an extension of the FOND planner PRP that allows it to find solutions for problems that contain actions with conditional effects.

PRP does not produce a universal plan, but rather a strong cyclic plan that takes the form of a partial policy mapping a subset of the states to an action. As such there are states that it *handles*, returning a next action to perform, and other states for which it does not. Our core algorithm, following the basic structure of PRP, is as follows:

1. Initialize *Open* and *Closed* lists of states handled by the incumbent policy P (*Open* initially contains only s_0);
2. Select and move a state s from *Open* to *Closed* such that,
   (a) If $P(s) = \perp$, run $\mathit{UpdatePolicy}(\langle \mathcal{V}, s, s_*, \mathcal{A} \rangle, P)$;
   (b) If $P(s) = a$, and $a \neq \perp$, add to *Open* every state in $\{\mathit{Prog}(s, a, \mathit{Eff}_i) \mid a = \langle pre_a, \mathit{Eff}_1, \ldots, \mathit{Eff}_k \rangle\} \setminus \mathit{Closed}$;
   (c) If *UpdatePolicy* failed in 2(a), process s as a *deadend*;
3. If *Open* is empty, return P. Else, repeat from step 2;

The procedure *UpdatePolicy* is a fundamental computational step of both the original and our extended PRP approach. Primarily, it involves (i) computing a weak plan via the all-outcomes determinization of $\langle \mathcal{V}, s, s_*, \mathcal{A} \rangle$, (ii) computing sufficient conditions for each suffix of the weak plan to achieve the goal, and (iii) adding the sufficient conditions, paired with the action at the start of the corresponding suffix, to the incumbent policy P. It is the computation of these condition-action pairs that is the major challenge in enabling PRP to handle conditional effects.

PRP has a number of other auxiliary components that are described in the original work (Muise, McIlraith, and Beck 2012; Muise 2014): (i) deadend detection, generalization, and avoidance, (ii) stopping conditions for weak plans, and (iii) conditions for stopping the simulation of action effects (replacing steps 2(a) - 2(c)). We generalized these components using the intuitions described below.

3.1 Condition-Action Pairs

A critical element in the efficiency of PRP is its ability to identify the subset of a state that is *relevant* to the validity of a weak plan that leads to the goal. This subset of the state, which is simply a partial state, is realized in the condition of the condition-action pairs that are used to generate the PRP policy. PRP was originally developed for

planning problems with no conditional effects. In such a setting, the conditions of condition-action pairs are equivalent to the states realized by regression (Waldinger 1977; Reiter 2001) of the goal through the actions in the weak plan. The regression rewriting operation identifies necessary and sufficient conditions for the plan suffix to be a valid plan. Regression in such a syntactically restricted setting leads to a unique partial state. When conditional effects are introduced, the regression can result in a *set* of (possibly) different partial states, reflecting the (possibly) different conditions under which execution of the action would lead to the successor state that is being rewritten. In general, these sets of partial states could be represented as a disjunction of conjunctions of literals (i.e., DNF). However, PRP requires the condition of the condition-action pair to be representable by a unique partial state. This unique partial state can be identified by filtering the regressed formula with respect to the state in which the action was originally applied when the weak plan was initially produced. This filtering results in sufficient conditions for the validity of the plan suffix. To avoid the computational overhead associated with using full regression, we instead aggregate the regression and filtering into a single efficient operation.

More precisely, the definition of regression over conditional effects in SAS$^+$, $Regr(s, a, Eff)$, corresponds to that defined in (Reiter 2001) with the straightforward accommodations for notation. As discussed above, we consider an operation PRIMF (***Pre-Im**age **F**iltering*) that combines both the regression and filtering. PRIMF utilizes the action effect and condition that was achieved by the action (as is the case with standard regression), as well as the state s_c that the action was applied in, simply termed the *context*.

Definition 1. Given partial states s and s_c, action $a = \langle pre_a, Eff_1, \ldots, Eff_n \rangle$ that achieved s from s_c, and associated effect $Eff = \{\langle cond_1, v_1, d_1 \rangle, \ldots, \langle cond_k, v_k, d_k \rangle\}$, we define PRIMF$(s, a, Eff, s_c)$ as the partial state p such that, for $v \in \mathcal{V}$:

$$p(v) = \begin{cases} s_c(v) & \text{if } v \in Support \\ \bot & \text{if } v \in Added \setminus Support \\ s(v) & \text{otherwise} \end{cases}$$

where,

$$Support = pre_a \cup \bigcup_{i=1\ldots n} \bigcup_{\langle cond, v, d \rangle \in Eff_i} cond$$

$$Added = \{v \mid \langle cond, v, d \rangle \in Eff \text{ and } s_c \models cond\}$$

Intuitively, *Support* consists of the variables mentioned in the condition of any conditional effect of a while *Added* consists of the variables the action made true given the context s_c and a particular action effect, *Eff*.

Proposition 1. The following properties follow from the definition of the PRIMF operator:

1. $s_c \models$ PRIMF(s, a, Eff, s_c)
2. PRIMF$(s, a, Eff, s_c) \models Regr(s, a, Eff)$
3. When there are no conditional effects PRIMF$(s, a, Eff, s_c) = Regr(s, a, Eff)$

These properties of PRIMF allow us to generalize PRP to handle conditional effects (c.f. Section 3.2), giving us the following theorem.

Theorem 1 (Soundness and Completeness)**.** Using PRIMF for the conditions computed by *UpdatePolicy*, our approach computes a strong cyclic plan iff such a plan exists.

That is, as far as correctness is concerned, we reap the same benefits as the original PRP work, albeit in a considerably more expressive setting. We now describe how PRP needs to be modified to accommodate the intuitions described above.

3.2 Integration with PRP

The PRIMF operator provides a means of computing a partial state that is a sufficient condition for a deterministic action to achieve another partial state. Wherever PRP originally used logical regression, we instead use the PRIMF operator. Unlike regression, PRIMF requires a context state to be defined. During the *UpdatePolicy* procedure, we naturally have a context state from the forward search procedure, and we use this state for the computation of PRIMF, yielding sufficient condition for the goal to be achieved by the plan suffix.

The original implementation of PRP also used regression when processing deadends (cf. step 2(c) from the above procedure). PRP would regress a deadend state through every action effect that could lead to the deadend. Unfortunately, we require a context state to perform the PRIMF operation. As such, here whenever a deadend is recorded we also record the action that led to the deadend along with the previous state, which provides a context for PRIMF. While slightly less general than the original, prohibiting the offending action in the context allows the planner to avoid reaching the deadend through similar means. For actions without conditional effects that could lead to the deadend, we resort to the original PRP method of regression as no context state is required.

4 Evaluation & Discussion

In this section we report on a preliminary evaluation of the newly implemented system to give a general sense of the planner's capability to solve problems with conditional effects. Because of the lack of planners capable of solving FOND planning problems with conditional effects, there are currently no appropriate benchmarks published. To remedy this, we introduce four domains that are modifications of existing benchmarks from the International Planning Competitions (IPC):[1] (1) Search and Rescue (**search**), (2) Sloppy Schedule (**schedule**), (3) Buggy Miconic (**miconic**), and (4) Tedious Triangle Tireworld (**tireworld**). The first three domains contain conditional effects with a complexity that makes compiling the actions into new ones without conditional effects too difficult using traditional means, while the final domain demonstrates how the new planner performs in domains containing needless conditional effects. We discuss the details of each domain below.

We conducted the experiment on a 3.4GHz CPU with time and memory limits of 30min and 1GB. To the best of our knowledge, Gamer is the only planner previously capable of solving non-deterministic problems with conditional effects (Kissmann and Edelkamp 2009). Unfortunately, Gamer is

[1] Planner source and benchmark sets can be found online at `http://www.haz.ca/research/prp/`

only capable of handling a simple form of conditional effects, prohibiting it from being used in all but one domain. Thus, we restrict our analysis to the new PRP system.

In Figure 1 we show both the policy size and the time to compile the policy for all of the problems in each of the four domains. Overall, we found that the performance of PRP was not adversely affected by the additions made to solve problems with conditional effects. The coverage for each domain is as follows: search (15/15), schedule (108/150), miconic (145/145), and tireworld (36/40). In what follows, we discuss each of the domains and how the planner performed.

Search and Rescue: The Search and Rescue domain originates from the IPC-6 Probabilistic track. The planning task requires the agent to fly to various locations and explore for a lost human who must be rescued. The action of "exploring" a location non-deterministically reveals whether or not a human is there and prohibits exploring that location again. As a result, no strong cyclic solution exists, but the probability of the optimal policy achieving the goal tends to 1 as the number of locations increases. We converted the probabilistic action effects into non-deterministic action effects in the same manner described in (Muise, McIlraith, and Beck 2012). This domain poses a challenge for PRP, and we chose to include it as an example of non-trivial conditional effects.

With the default planner settings, the problems in the Search and Rescue domain become very difficult to solve – the planner spends the majority of time trying to prove that a strong cyclic solution does not exist. The dead-end generalization used by PRP is not applicable for this domain, and this demonstrates a promising area of improvement for that mechanism. We limited the search time for the planner on this domain to 10 seconds, and used the best policy found by that point. No better policy was produced beyond 10 seconds, and PRP would simply timeout trying to prove no strong cyclic solution existed. The best policy that was produced by PRP, however, coincided precisely with the optimal policy in the probabilistic setting. In other words, PRP successfully computed the best policy it could hope to find, but failed to prove no strong cyclic solution existed within a reasonable amount of time. It should be emphasized that even if PRP was permitted to run to completion, there is no guarantee from the planner that the policy it finds is the best possible for the domain, as no strong cyclic solution exists.

Sloppy Schedule: The Sloppy Schedule domain is a modification of the Schedule World domain in the IPC-2 competition. The problem involves processing different physical components in a workshop setting through a series of actions – painting, sanding, cutting, etc. Non-determinism was introduced to the domain in two places: (1) to mimic a "sloppy" operator, in which many of the operations change the state of the component being processed (e.g., polishing could cause the paint to wear off), and (2) other machines in the workshop may become occupied while a component is being processed. This domain was selected due to the significant number of complicated conditional effects it contains.

The planner finds a very compact policy for the Sloppy Schedule domain in the problems it was able to solve. The

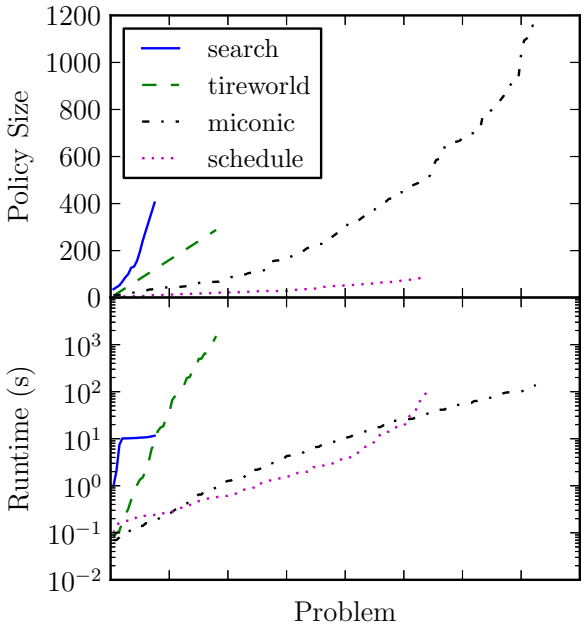

Figure 1: Policy size and compilation time for all problems in the four domains (x-axis sorted on problem difficulty).

reduction in coverage is due to the FF heuristic used at the core of PRP which is not able to solve the determinization effectively. This may be remedied by modifying PRP to use a more powerful heuristic, such as LM-cut (Helmert and Domshlak 2009), for the weak planning process.

Buggy Miconic: We based the Buggy Miconic domain on the Miconic domain from the IPC-2 competition. In this domain, an elevator is modelled and a *stop* action transfers people on or off an elevator depending on whether or not they must be picked up or dropped off respectively. Two additional actions allow the elevator to change floors in either direction. The non-determinism is added to the domain by changing the *up* and *down* actions so that the elevator does not always arrive at the intended floor. It may stop early or go beyond the target destination. This non-determinism models the possibility where human users could choose the wrong floor or press a button by mistake, and allows the planner to compensate for this undesired outcome.

The Miconic domain represents one of the largest challenges for the modified version of PRP as far as conditional effects are concerned – the *stop* action mentions almost every variable in the precondition which causes PRP to behave as if it is planning with complete states. Nevertheless, PRP is still capable of solving every problem in the domain, albeit with a steadily increasing policy size.

Tedious Triangle Tireworld: The Tedious Triangle Tireworld domain is a modification of the Triangle Tireworld domain in the IPC-6 FOND track. In this domain a car must navigate a map with a possibility of getting a flat tire each time the car changes locations. For this domain, we replaced some of the preconditions of the non-deterministic actions with conditional effects. For example, the car can always try to drive from one location to another, but it will only succeed

if it does not already have a flat tire. Ideally, the modified version of PRP should perform just as well in the Tedious Triangle Tireworld domain as the original PRP planner on the Triangle Tireworld domain.

The performance of the modified PRP on the Tedious Triangle Tireworld is almost identical to PRP on the original domain – the coverage drops from 38 problems to 36 for the modified domain. This demonstrates that there is very little impact resulting from modelling action behaviour as conditional effects in lieu of preconditions. The slight drop in coverage is to be expected, as the Tedious version has a much larger branching factor, which slows down the planning process for the determinized planning problems. This is the one domain in which Gamer is capable of solving instances, but the coverage is only 5 problems out of 40.

5 Concluding Remarks

We presented an extension to the PRP planner that enables it to solve FOND planning problems in domains where actions may contain conditional effects. Our modifications to PRP were systematic: a number of formal and computational steps were necessary to benefit from all of the features that make PRP a competitive solver. Most significantly, we maintained the soundness and completeness of the planner. Additionally, we conducted a preliminary investigation into the strengths and weaknesses of the newly created planner and found that it is highly capable of solving non-deterministic problems with conditional effects. As is to be expected, certain domains can make the planning process more difficult in this more expressive setting: we discovered a domain where the deadend generalization is insufficient for the problems to be optimally solved, and another that requires a more powerful heuristic for the determinization component. We think these are worthy of further study, and computational schemes that may work here are left for the immediate future. We also hope to investigate a refinement of PRIMF that more closely captures the result of regression. Alternatively, we may return to the computation of policy conditions via regression, accompanied by further modifications to PRP. The newly introduced planner represents an important increase in the scope of problems that can be solved by non-deterministic planning techniques. This development is significant not only for FOND planning but for emerging applications that can exploit FOND planning in their computational core (e.g., (Ramírez, Yadav, and Sardiña 2013; Patrizi, Lipovetzky, and Geffner 2013)).

Acknowledgements We gratefully acknowledge funding from the Natural Sciences and Engineering Research Council of Canada.

References

Cimatti, A.; Pistore, M.; Roveri, M.; and Traverso, P. 2003. Weak, strong, and strong cyclic planning via symbolic model checking. *Artificial Intelligence* 147(1):35–84.

Fu, J.; Ng, V.; Bastani, F. B.; and Yen, I.-L. 2011. Simple and fast strong cyclic planning for fully-observable nondeterministic planning problems. In *Proceedings of the 22nd International Joint Conference On Artificial Intelligence*, 1949–1954.

Helmert, M., and Domshlak, C. 2009. Landmarks, critical paths and abstractions: What's the difference anyway? In *Proceedings of the 19th International Conference on Automated Planning and Scheduling*.

Helmert, M. 2009. Concise finite-domain representations for PDDL planning tasks. *Artificial Intelligence* 173(5-6):503–535.

Kissmann, P., and Edelkamp, S. 2009. Solving fully-observable non-deterministic planning problems via translation into a general game. In *Proceedings of the 32nd Annual German Conference on Advances in Artificial Intelligence*, KI'09, 1–8. Berlin, Heidelberg: Springer-Verlag.

Kuter, U.; Nau, D.; Reisner, E.; and Goldman, R. P. 2008. Using classical planners to solve nondeterministic planning problems. In *Proceedings of the 18th International Conference on Automated Planning and Scheduling*, 190–197.

Mattmüller, R.; Ortlieb, M.; Helmert, M.; and Bercher, P. 2010. Pattern database heuristics for fully observable nondeterministic planning. In *Proceedings of the 20th International Conference on Automated Planning and Scheduling*, 105–112.

Muise, C.; McIlraith, S. A.; and Beck, J. C. 2012. Improved Non-Deterministic Planning by Exploiting State Relevance. In *Proceedings of the 22nd International Conference on Automated Planning and Scheduling*, The 22nd International Conference on Automated Planning and Scheduling.

Muise, C. 2014. *Exploiting Relevance to Improve Robustness and Flexibility in Plan Generation and Execution*. Ph.D. Dissertation, University of Toronto.

Patrizi, F.; Lipovetzky, N.; and Geffner, H. 2013. Fair LTL synthesis for non-deterministic systems using strong cyclic planners. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence*, 2343–2349.

Ramírez, M.; Yadav, N.; and Sardiña, S. 2013. Behavior composition as fully observable non-deterministic planning. In *Proceedings of the 23rd International Conference on Automated Planning and Scheduling*, 180–188.

Reiter, R. 2001. *Knowledge in action: logical foundations for specifying and implementing dynamical systems*. The MIT Press.

Waldinger, R. 1977. Achieving several goals simultaneously. In *Machine Intelligence 8*. Edinburgh, Scotland: Ellis Horwood. 94–136.