Flexible FOND Planning with Explicit Fairness Assumptions*

Ivan D. Rodriguez,[1] Blai Bonet,[1] Sebastian Sardiña,[2] Hector Geffner[1,3]
[1]Universitat Pompeu Fabra, Barcelona, Spain
[2]RMIT University, Melbourne, Australia
[3]Institució Catalana de Recerca i Estudis Avançats (ICREA)
ivandanielra@gmail.com, bonetblai@gmail.com,
sebastian.sardina@rmit.edu.au, hector.geffner@upf.edu

Abstract

We consider the problem of reaching a propositional goal condition in fully-observable non-deterministic (FOND) planning under a general class of fairness assumptions that are given explicitly. The fairness assumptions are of the form A/B and say that state trajectories that contain infinite occurrences of an action a from A in a state s and finite occurrence of actions from B, must also contain infinite occurrences of action a in s followed by each one of its possible outcomes. The infinite trajectories that violate this condition are deemed as *unfair,* and the solutions are policies for which *all the fair trajectories reach a goal state.* We show that strong and strong-cyclic FOND planning, as well as QNP planning, a planning model introduced recently for generalized planning, are all special cases of FOND planning with fairness assumptions of this form which can also be combined. FOND$^+$ planning, as this form of planning is called, combines the syntax of FOND planning with some of the versatility of LTL for expressing fairness constraints. A new planner is implemented by reducing FOND$^+$ planning to answer set programs, and the performance of the planner is evaluated in comparison with FOND and QNP planners, and LTL synthesis tools.

Introduction

FOND planning is planning with fully observable, non-deterministic state models specified in compact form where a goal state is to be reached (Cimatti et al. 2003). In its most common variant, strong-cyclic planning, one is interested in policies that reach states from which the goal can be reached following the policy (Cimatti, Roveri, and Traverso 1998a; Daniele, Traverso, and Vardi 1999). In another common variant, strong planning (Cimatti, Roveri, and Traverso 1998b), one is interested in policies that reach a goal state in a bounded number of steps. Each form of FOND planning is adequate under a suitable *fairness* assumption; in the case of strong planning, that non-determinism is adversarial (or "unfair"); in the case of strong-cyclic planning, that non-determinism is fair, in that none of the possible outcomes of a non-deterministic action can be skipped forever.

*A longer version of this paper with all the proofs is available (Rodriguez et al. 2021).

FOND planning has become increasingly important as a way of solving other types of problems such as *probabilistic (MDP) planning*, where actions have a probabilistic effect on states (Bertsekas and Tsitsiklis 1996; Geffner and Bonet 2013), *LTL planning*, where goals to be reached are generalized to temporal conditions that must be satisfied possibly by plans with cycles (Calvanese, De Giacomo, and Vardi 2002; Camacho, Bienvenu, and McIlraith 2019; Aminof et al. 2019), and *generalized planning*, where plans are not for single instances but for collections of instances (Srivastava, Immerman, and Zilberstein 2011; Hu and De Giacomo 2011), and they can be obtained from suitable abstractions encoded as QNP planning problems (Srivastava et al. 2011; Bonet and Geffner 2020).

A critical limitation of strong, strong-cyclic, and QNP planners, is that the fairness assumptions are implicit in their models and solvers, and as a result, cannot be combined. These combinations, however, are often needed (Camacho and McIlraith 2016; Ciolek et al. 2020), and indeed, a recent FOND planner handles combinations of fair and adversarial actions in what is called *Dual FOND planning* (Geffner and Geffner 2018). In this work, we go beyond this integration by also enabling the representation and combination of the conditional fairness assumptions that underlie QNP planning. This is achieved by extending FOND planning with a general class of fairness assumptions that are given explicitly as part of the problem. The fairness assumptions are pairs A/B of sets of actions A and B that say that state trajectories that contain infinite occurrences of actions a from A in a state s, and finite occurrences of actions from B, must also contain infinite occurrences of action a in the state s followed by each one of its possible outcomes. The infinite trajectories that violate this condition are regarded as *unfair.* The solutions of a FOND problem with conditional fairness assumptions of this type, called a FOND$^+$ problem, are *the policies for which all fair state trajectories reach the goal.*

We show that strong, strong-cyclic, and QNP planning, are all special cases of FOND$^+$ planning where the fairness assumptions underlying these models can be combined. FOND$^+$ planning extends the syntax and semantics of FOND planning with some of the versatility of the LTL language for expressing fairness constraints. The conditional

fairness assumptions A/B correspond to the LTL formulas $(\Box\Diamond(s\wedge a)\wedge(\neg\Box\Diamond\bigvee_{b\in B} b))\supset(\bigwedge_i \Box\Diamond(a\wedge s\wedge\bigcirc E_i)$, one for each action $a \in A$, each state s, and each possible outcome E_i of the action a, where s stands for the conjunction of literals that s makes true. However, unlike LTL synthesis and planning that are 2EXP-Complete (Pnueli and Rosner 1989; Camacho, Bienvenu, and McIlraith 2019; Aminof, De Giacomo, and Rubin 2020), FOND$^+$ planning is in NEXP (non-deterministic exponential time).

A planner for FOND$^+$ is obtained by reducing FOND$^+$ planning over the explicit state space to an elegant answer set program (ASP), a convenient and high-level alternative to SAT (Brewka, Eiter, and Truszczyński 2011; Lifschitz 2019; Gebser et al. 2012), using the facilities provided by the CLINGO ASP solver (Gebser et al. 2019). The performance of this ASP-based planner is evaluated in comparison with FOND and QNP planners, and LTL synthesis tools.

The paper is organized as follows. We review first strong and strong-cyclic FOND planning, and QNP planning. We introduce then FOND$^+$ planning, where the assumptions underlying these models are stated explicitly and combined, and present a description of the ASP-based FOND$^+$ planner, an empirical evaluation, and a discussion.

FOND Planning

A FOND model is a tuple $M = \langle S, s_0, S_G, Act, A, F\rangle$, where S is a finite set of states, $s_0 \in S$ is the initial state, $S_G \subseteq S$ is a non-empty set of goal states, Act is a set of actions, $F(a, s)$ is the set of successor states when action a is executed in state s, and $A(s) \subseteq Act$ is the set of actions applicable in state s, such that $a \in A(s)$ iff $F(a, s) \neq \emptyset$. A FOND problem P is a compact description of a FOND model $M(P)$ in terms of a finite set of atoms, so that the states s in $M(P)$ correspond to truth valuations over the atoms, represented by the set of atoms that are true. The standard syntax for FOND problems is a simple extension of the STRIPS syntax for classical planning. A FOND problem is a tuple $P = \langle At, I, Act, G\rangle$ where At is a set of atoms, $I \subseteq At$ is the set of atoms true in the initial state s_0, G is the set of goal atoms, and Act is a set of actions with atomic preconditions and effects. If E_i represents the set of positive and negative effects of an action in the classical setting, action effects in FOND planning can be deterministic of the form E_i, or non-deterministic of the form *oneof*$(E_1, \ldots, E_n)$.

A policy π for a FOND problem P is a partial function mapping *non-goal* states into actions. A policy π for P defines a set of, possibly infinite, compatible state trajectories $s_0, s_1, s_2, \ldots$, also called **π-trajectories**, where $s_{i+1} \in F(a_i, s_i)$ and $a_i = \pi(s_i)$ for $i \geq 0$. A trajectory τ compatible with π is *maximal* if it is infinite, or is finite of the form $\tau = s_0, \ldots, s_n$, for some $n \geq 0$, and either s_n is the first state in the sequence being a goal state, $\pi(s_n) \notin A(s_n)$ (i.e., the action prescribed at s_n is not applicable), or $\pi(s_n) = \bot$ (i.e., no action is prescribed). Likewise, the policy π reaches a state s if there is a π-trajectory $s_0, \ldots, s_n$ where $s = s_n$, and π reaches a state s' from a state s if there is a π-trajectory $s_0, \ldots, s_n$ where $s = s_i$ and $s' = s_j$ for $0 \leq i \leq j \leq n$. A state s is **recurrent** in trajectory τ if it appears an infinite number of times in τ. The strong and strong-cyclic solutions or policies are usually defined as follows:

Definition 1 (Solutions)**.** *A policy π is a **strong solution** for a FOND problem P if all the maximal π-trajectories reach a goal state, and it is a **strong-cyclic solution** if π reaches a goal state from any state reached by π.*

The strong solutions correspond also to the strong-cyclic solutions that are acyclic; namely, where the policies π do not give rise to π-trajectories that can visit a state more than once. Alternatively, strong and strong-cyclic solutions can be understood in terms suitable notions of fairness that establish which π-trajectories are deemed possible. If we say that *a policy π solves problem P when all the **fair** π-trajectories reach the goal*, then in strong planning, all π-trajectories are deemed fair, while in strong-cyclic planning, all π-trajectories are deemed fair *except* those containing a recurrent state s that is followed a finite number of times by a successor $s' \in F(\pi(s), s)$.

In order to make this alternative "folk" characterization of strong and strong-cyclic planning explicit, let us say that all the actions in strong FOND planning are **adversarial** (or "unfair"), and that all the actions in strong-cyclic FOND planning are **fair**. The state trajectories that are deemed **fair** in each setting can then be expressed as follows:

Definition 2. *If all the actions are **adversarial**, all π-trajectories are **fair**. If all the actions are **fair**, a π-trajectory τ is **fair** iff states s that occur an infinite number of times in τ, are followed an infinite number of times by each possible successor s' of s given π, $s' \in F(\pi(s), s)$.*

Provided with these notions of fairness, strong and strong-cyclic solutions can be characterized equivalently as:

Theorem 3. *A policy π is a strong (resp. strong-cyclic) solution of a FOND problem P iff all the **fair** trajectories compatible with π in P reach the goal, under the assumption that all actions are **adversarial** (resp. **fair**).*

Methods for computing strong and strong-cyclic solutions for FOND problems have been developed based on OBDDs (Cimatti et al. 2003), explicit forms of AND/OR search (Mattmüller et al. 2010), classical planners (Muise, McIlraith, and Beck 2012), and SAT (Chatterjee, Chmelík, and Davies 2016). Some of these planners actually handle a **combination** of fair and adversarial actions, in what is called Dual FOND planning (Geffner and Geffner 2018).

QNP Planning

Qualitative numerical planning problems (QNPs) were introduced by Srivastava et al. (2011) as a model for generalized planning, that is, planning for multiple classical instances at once. QNPs have been used since in other works (Bonet et al. 2017; Bonet, Frances, and Geffner 2019) and have been analyzed in depth by Bonet and Geffner (2020).

The syntax of QNPs is an extension of STRIPS problems $P = \langle At, I, O, G\rangle$ with negation where At is a set of ground (boolean) atoms, I is a maximal consistent set of literals from At describing the initial situation, G is a set of literals describing the goal situation, and O is a set of (ground) actions with precondition and effect literals. A

QNP $Q = \langle At, V, I, O, G \rangle$ extends a STRIPS problem with a set V of *numerical variables* X that can be decremented or incremented *qualitatively*; i.e., by indeterminate positive amounts, without making the variables negative. A numerical variable X can appear in action effects as $X\uparrow$ (increments) and $X\downarrow$ (decrements),while literals of the form $X = 0$ or $X > 0$ (an abbreviation of $X \neq 0$) can appear everywhere else (initial situation, preconditions, and goals). The literal $X > 0$ is a precondition of all actions with $X\downarrow$ effects.

A simple example of a QNP is $Q = \langle At, V, I, O, G \rangle$ with $At = \{p\}$, $V = \{n\}$, $I = \{\neg p, n > 0\}$, $G = \{n = 0\}$, and actions $O = \{a, b\}$ given by

$$a = \langle p, n > 0; \neg p, n\downarrow \rangle \quad \text{and} \quad b = \langle \neg p; p \rangle$$

where $\langle C; E \rangle$ denotes an action with preconditions C and effects E. Thus action a decrements n and negates p that is a precondition of a, and b restores p. This QNP represents an abstraction of the problem of clearing a block x in Blocksworld instances with stack/unstack actions that include a block x. The numerical variable n stands for the number of blocks above x, and the boolean variable p stands for the robot gripper being empty. A policy π that solves Q can be expressed by the rules:

$$\textbf{if } p \text{ and } n > 0, \textbf{ do } a \quad \text{and} \quad \textbf{if } \neg p \wedge n > 0, \textbf{ do } b\,.$$

A key property of QNPs is that while numerical planning is undecidable (Helmert 2002), qualitative numerical planning is not. Indeed, a sound and complete, two-step method for solving QNPs was formulated by Srivastava et al. (2011): the QNP Q is converted into a standard FOND problem $P = T_D(Q)$ and its (strong-cyclic) solution is checked for termination. The QNP solutions are in correspondence with the **strong-cyclic plans** of the direct translation $P = T_D(Q)$ that **terminate.** Moreover, since the number of policies that solve P is finite, and the termination of each can be verified in finite time, plan existence for QNPs is decidable. More recent work has shown that the complexity of QNP planning is the same as that of FOND planning by introducing a **polynomial reduction** from the former into the latter, and another in the opposite direction (Bonet and Geffner 2020).

We do not need to get into the formal details of QNPs but it is useful to review the direct translation T_D of a QNP Q into a FOND problem $P = T_D(Q)$, and the notion of termination (Srivastava et al. 2011). Concretely, the translation T_D replaces each numerical variable n by a boolean atom p_n that stands for the (boolean) expression $n = 0$. Then, occurrences of the literal $n = 0$ in the initial situation, action preconditions, and goals are replaced by p_n, while occurrences of the literal $n > 0$ in the same contexts are replaced by $\neg p_n$. Likewise, effects $n\uparrow$ are replaced by effects $\neg p_n$, and effects $n\downarrow$ are replaced by non-deterministic effects *oneof*$(p_n, \neg p_n)$. Actions in the FOND problem $P = T_D(Q)$ with effects $\neg p_n$ (i.e., $n > 0$) are said to "increment n," while actions with effects *oneof*$(p_n, \neg p_n)$ (i.e., either $n > 0$ or $n = 0$) are said to "decrement n," even if there are no numerical variables in P but just boolean variables. This information needs to be preserved in the translation $P = T_D(Q)$, as the semantics of P is not the semantics of FOND problems as assumed by strong or strong-cyclic planners.

Termination and SIEVE

A policy π for the FOND problem $P = T_D(Q)$ is said to **terminate** if all the state trajectories in P that are compatible with the policy π and with the fairness assumptions underlying the QNP Q, are finite. Termination is the result of the absence of cycles in the policy that could be traversed forever. The latter arises when a cycle includes an action that decrements a numerical variable and none that increments it. Since numerical variables cannot become negative such cycles eventually terminate.

The procedure called SIEVE (Srivastava et al. 2011) provides a **sound and complete termination test** that runs in time that is polynomial in the number of states reached by the policy. SIEVE can be understood as an efficient implementation of the following procedure that operates on a policy graph $G(P, \pi)$ induced by the FOND problem P and the policy π, where the nodes are the states s that can be reached in P via the policy π, and the edges correspond to the state transitions (s, s') that are possible given the policy π (i.e., $s' \in F(\pi(s), s)$).

Starting with the graph $\mathcal{G} = G(P, \pi)$, SIEVE iteratively removes edges from $\mathcal{G}$ until $\mathcal{G}$ becomes acyclic or does not admit further removals. In each iteration, an edge (s, s') is removed from $\mathcal{G}$ if $\pi(s)$ is an action that decrements a variable x that is not incremented along any path in $\mathcal{G}$ from s' back to s. SIEVE **accepts** the policy π iff SIEVE renders the resulting graph $\mathcal{G}$ acyclic. It can be shown that the resulting graph $\mathcal{G}$ is well defined (i.e., it is the same independently of the order in which edges are removed), and that SIEVE removes an edge (s, s') when it cannot be traversed by the policy an infinite number of times.

It is useful to capture the logic of SIEVE in terms of an **inductive definition** that considers states instead of edges:

Definition 4 (QNP Termination)**.** *Let π be a policy for the FOND problem $P = T_D(Q)$ associated with the QNP Q. The policy π **terminates** in P iff every state s that is reachable by π in P terminates, where a state s **terminates** iff:*[1]

1. *there is no cycle on node s (i.e., no path from s to itself),*
2. *every cycle on s contains a state s' that **terminates**, or*
3. *$\pi(s)$ decrements a variable x, and every cycle on s containing a state s' for which $\pi(s')$ increments x, also contains a state s'' that **terminates**.*

Theorem 5. *Let Q be a QNPs and π a policy. Then,* SIEVE ***accepts*** *the policy graph $G(P, \pi)$ iff policy π **terminates** in P, where $P = T_D(Q)$.*

Since solutions to QNPs Q are known to be the strong-cyclic policies of the FOND problem $P = T_D(Q)$ that are accepted by SIEVE (Srivastava et al. 2011; Bonet and Geffner 2020), the solutions for Q can also be expressed as:

Theorem 6. *A policy π is a solution to a QNP Q iff π is a strong-cyclic solution of $P = T_D(Q)$ that terminates.*

[1] This inductive definition and the ones below imply that there is a **unique sequence** of state subsets $S_0, S_1, \ldots, S_k$ such that S_{i+1} is S_i augmented with all the states that can be added to S_i when assuming that the only terminating states are those in S_i.

The characterization that results from this theorem has been used to verify QNP solutions but not for *computing* them. Indeed, the only available complete QNP planner is based on a polynomial reduction of QNP planning into strong-cyclic FOND planning that avoids the termination test (Bonet and Geffner 2020).

FOND$^+$ Planning

In this section, we move from strong, strong-cyclic, and QNP planning to the FOND$^+$ setting where the fairness assumptions underlying these models can be explicitly stated and combined. A **FOND$^+$ planning problem** $P_c = \langle P, C\rangle$ is a FOND problem P extended with a set C of fairness assumptions:

Definition 7. *A FOND$^+$ problem $P_c = \langle P, C\rangle$ is a FOND problem P extended with a set C of **(conditional) fairness assumptions** of the form A_i/B_i, $i = 1, \ldots, n$ and where each A_i is a set of **non-deterministic actions** in P, and each B_i is a set of actions in P disjoint from A_i.*

The fairness assumptions play no role in constraining the state trajectories that are possible by following a policy π, the so-called π-trajectories:

Definition 8. *A state trajectory compatible with a policy π for the FOND$^+$ problem $P_c = \langle P, C\rangle$ is a state trajectory that is compatible with π in the FOND problem P.*

However, while in strong and strong-cyclic FOND planning all actions are considered as adversarial and fair, respectively, in the FOND$^+$ setting, each action is labeled fair or unfair depending on the assumptions in C and the trajectory where the action occurs. We define what it means for an action $a = \pi(s)$ to behave "fairly" in a recurrent state s of an infinite π-trajectory as follows:

Definition 9. *The occurrence of the action $\pi(s)$ in a recurrent state s of a π-trajectory τ associated with the FOND$^+$ problem $P_c = \langle P, C\rangle$ is **fair** if for some fairness assumption $A/B \in C$, it is the case that $\pi(s) \in A$ and all the actions in B occur finitely often in τ.*

The meaning of a conditional fairness assumption A/B is that the actions $a \in A$ can be assumed to be **fair** in any recurrent state s of a π-trajectory τ, provided that the condition on B holds in τ; namely, that actions in B do *not* occur infinitely often in τ. Otherwise, if any action in B occurs infinitely often in τ, then a is said to be **unfair** or **adversarial**. Once actions $\pi(s)$ occurring in recurrent states s are "labeled" in this way, the standard notion of fair trajectories (Definition 2) extends naturally to FOND$^+$ problems:

Definition 10. *A **π-trajectory** τ for a FOND$^+$ problem $P_c = \langle P, C\rangle$ is **fair** if for every recurrent state s in τ where the action $\pi(s)$ is **fair** and every possible successor s' of s due to action $\pi(s)$ (i.e., $s' \in F(\pi(s), s)$), state s is immediately followed by state s' in τ an infinite number of times.*

The solution of FOND$^+$ problems can then be expressed in a standard way as follows:

Definition 11 (Solutions)**.** *A policy π **solves** the FOND$^+$ problem $P_c = \langle P, C\rangle$ if the maximal π-trajectories that are **fair** reach the goal.*

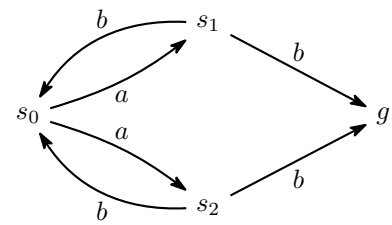

Figure 1: Example model for FOND problem P with 4 states, non-deterministic actions a and b, and goal state g.

A number of observations can be drawn from these definitions. Let us say that one wants to model a non-deterministic action a whose behavior is **fair** in that it always displays all its possible effects infinitely often in every recurrent state s such that $\pi(s) = a$. To do so, we consider a fairness constraint A/B in C such that $a \in A$ and B is empty. On the other hand, to model an **adversarial** action b, one whose behavior is not fair (may not yield all its effects infinitely often in a recurrent state s with $\pi(s) = b$), we do not include b in any set A. This immediately suggests the way to capture standard strong and strong-cyclic planning as special forms of FOND$^+$ planning:

Theorem 12. *The **strong solutions** of a FOND problem P are the solutions of the FOND$^+$ problem $P_c = \langle P, \emptyset\rangle$.*

Theorem 13. *The **strong-cyclic solutions** of a FOND problem P are the solutions of the FOND$^+$ problem $P_c = \langle P, \{A/\emptyset\}\rangle$, where A is the set of all the non-deterministic actions in P.*

Similarly, QNP problems are reduced to FOND$^+$ problems in a direct way, in this case, making use of both the head A and the condition B in the fairness assumptions A/B in C:

Theorem 14. *The solutions of a **QNP problem** Q are the solutions of the FOND$^+$ problem $P_c = \langle P, C\rangle$ where $P = T_D(Q)$ and C is the set of fairness assumptions A_i/B_i, one for each numerical variable x_i in Q, such that A_i contains all the actions in P that decrement x_i, and B_i contains all the actions in P that increment x_i.*

Example

By explicitly stating the fairness assumptions underlying strong, strong-cyclic, and QNP planning, FOND$^+$ planning integrates these planning models as well. We illustrate the new possibilities with an example.

Let P be a FOND problem with state set $\{s_0, s_1, s_2, g\}$, two non-deterministic actions a and b, initial and goal states being s_0 and g, respectively. Action a can only be applied in state s_0, leading to either s_1 or s_2, whereas action b can be applied only in s_1 and s_2, leading, in both cases, to either s_0 or g; see Figure 1. The FOND problem P admits a single policy, namely, $\pi(s_0) = a$ and $\pi(s_1) = \pi(s_2) = b$, which we analyze in the context of different FOND$^+$ problems $P_i = \langle P, C_i\rangle$ that can be built on top of P using different sets of fairness assumptions C_i. For convenience, in the sets C_i, we use a/b to denote the fairness assumption $\{a\}$ / $\{b\}$,

and a to denote the assumption $\{a\}/\emptyset$. The marks '✓' and '✗' express that the policy π solves or does not solve, resp., the FOND$^+$ problem P_i, where C_i is:

✗ $C_1 = \{\}$; a and b are adversarial.

✓ $C_2 = \{a, b\}$; a and b are fair.

✗ $C_3 = \{a\}$; a is fair and b is adversarial.

✓ $C_4 = \{b\}$; b is fair and a is adversarial.

✗ $C_5 = \{a/b\}$; a is conditionally fair on b; b adversarial.

✗ $C_6 = \{a, b/a\}$; QNP like: $a : x_1\downarrow, x_2\uparrow$ and $b : x_2\downarrow$.

✓ $C_7 = \{b, a/b\}$; QNP like: $b : x_1\downarrow, x_2\uparrow$ and $a : x_2\downarrow$.

✗ $C_8 = \{a/b, b/a\}$; QNP like: $a : x_1\downarrow, x_2\uparrow$ and $b : x_2\downarrow, x_1\uparrow$.

The subtle cases are the last four. The policy π does not solve P_5 because there are trajectories like $\tau = s_0, s_1, s_0, s_2, s_0, s_1, s_0, \ldots$ that are fair but do not reach the goal. The reason is that while $a/b \in C_5$, the occurrences of the action $a = \pi(s_0)$ in the recurrent state s_0 in τ are not fair. Thus, both a and b have an adversarial semantics in τ. The policy π does not solve P_6 either, because in the same trajectory τ, the action a is fair in s_0 as $a \in C_6$ but b is not fair in either s_1 or s_2, as the assumption b/a is in C_6 but a occurs infinitely often in τ. As a result, τ is fair but non-goal reaching in P_6. The situation is different in P_7, where b is fair and a is unfair. Here, τ is unfair, as any other trajectory in which some or all the states s_0, s_1, and s_2 occur infinitely often. This is because b being fair in s_1 and s_2 means that the transitions (s_1, g) and (s_2, g) cannot be skipped forever, and the goal must be reached eventually. Finally, in P_8, the trajectory τ becomes fair again, as both a and b are adversarial in τ.

Termination and SIEVE$^+$ for FOND$^+$

We now consider the computation of policies for FOND$^+$ problems. Initially, we look for a procedure to verify if a policy π solves a problem $P_c = \langle P, C\rangle$, and then transform this *verification procedure* into a *synthesis procedure*.

The solutions for FOND$^+$ problems are policies that **terminate in the goal**, a termination condition that combines and goes beyond the solution concept for QNPs that only requires goal reachability (strong-cyclicity) and termination (finite trajectories). The termination condition for FOND$^+$ planning can be expressed as follows:

Definition 15 (FOND$^+$ termination)**.** *Let π be a policy for the FOND$^+$ problem P_c=$\langle P, C\rangle$. State s in P **terminates** iff*

1. *s is a goal state,*
2. *s is **fair** and some state $s' \in F(\pi(s), s)$ **terminates**, or*
3. *s is **not fair**, all states $s' \in F(\pi(s), s)$ **terminate**, and $F(\pi(s), s)$ is non-empty.*

*where s is **fair** if for some A_i/B_i in C, $\pi(s) \in A_i$, and every path that connects s to itself and that contains a state s' with $\pi(s') \in B_i$, also contains a state s'' that **terminates**.*

FOND$^+$ termination expresses a procedure similar to SIEVE, that we call SIEVE$^+$, that keeps labeling states s as terminating (the same as removing all edges from s in the policy graph) until no states are left or no more states can be labeled. The key difference with SIEVE is that the removals are done backward from the goal as captured in Definition 15. This is strictly necessary for SIEVE$^+$ to be a sound and complete procedure for FOND$^+$ problems:

Theorem 16. *A policy π solves the FOND$^+$ problem $P_c = \langle P, C\rangle$ iff all the states s that are reachable by π **terminate** according to Definition 15.*

The solutions to FOND$^+$ problems cannot be characterized as those of QNPs, as policies that are strong-cyclic and terminating in the sense that the policy cannot traverse edges in the policy graph forever. The policy π for the example P_5 is indeed strong-cyclic and terminating in this sense, but as shown above, it does not solve P_5. The policy terminates because the action a cannot be done forever, but it does not terminate in a goal state. In QNPs, this cannot happen, as strong-cyclic policies that are terminating, always terminate in a goal state.

FOND$^+$ and Dual FOND Planning

FOND$^+$ planning subsumes Dual FOND planning where fair and adversarial actions can be combined. In order to show that, let us first recall the latter:

Definition 17 (Geffner and Geffner, 2018)**.** *A Dual FOND problem is a FOND problem where the non-deterministic actions are labeled as either **fair** or **adversarial**. A policy π **solves** a Dual FOND problem P iff for all reachable state s, $\pi(s)\in A(s)$, and there is a function d from **reachable states** into $\{0, \ldots, |S|\}$ such that 1) $d(s)$=0 for goal states, 2) $d(s')<d(s)$ for **some** $s' \in F(\pi(s), s)$ if $\pi(s)$ is fair, and 3) $d(s')<d(s)$ for **all** $s' \in F(\pi(s), s)$ if $\pi(s)$ is adversarial.*

For showing that this semantics coincides with the semantics of a suitable fragment of FOND$^+$ planning, let us recast this definition as a termination procedure:

Definition 18 (Dual FOND termination)**.** *Let π be a policy for the Dual FOND problem P. A state s in P **terminates** iff*

1. *s is a goal state,*
2. *$\pi(s)$ is **fair** and some $s' \in F(\pi(s), s)$ **terminates**, or*
3. *$\pi(s)$ is **adversarial**, all states $s' \in F(\pi(s), s)$ **terminate**, and $F(\pi(s), s)$ is non-empty.*

Theorem 19. *π is a solution to a Dual FOND problem P iff for every non-goal state s reachable by π, $\pi(s) \in A(s)$ and s terminates according to Definition 18.*

The only difference between the termination for Dual FOND and the one for FOND$^+$ (Def. 15) is that in the former the fair and adversarial labels are given, while in the latter they are a function of the explicit fairness assumptions and policy. It is easy to show however that Dual FOND problems correspond to the class of FOND$^+$ problems with conditional fairness assumptions A/B with empty B:

Theorem 20. *A policy π solves a Dual FOND problem P' iff π solves the FOND$^+$ problem $P_c = \langle P, C\rangle$ where P is like P' without the action labels, and $C = \{A/B\}$ where A contains all the actions labeled as fair in P', and B is empty.*

FOND-ASP: An ASP-based FOND$^+$ Planner

The characterization of FOND$^+$ planning given in Theorem 16 allows for a transparent and direct implementation of a sound and complete FOND$^+$ planner. For this, the planner *hints* a policy π and then each state reachable by π is checked for termination using Definition 15. The problem of looking for a policy that satisfies this restriction can be expressed in SAT, although we have found it more convenient to express it as an **answer set program**, a convenient and high-level alternative to SAT (Brewka, Eiter, and Truszczyński 2011; Lifschitz 2019; Gebser et al. 2012), using the facilities provided by CLINGO (Gebser et al. 2019).

The code for the back-end of the ASP-based FOND$^+$ planner is shown in Figure 2. The front-end of the planner, not shown, parses an input problem $P_c = \langle P, C\rangle$ and builds a *flat representation* of P_c in terms of a number of ground atoms that are shown in capitalized predicates in the figure. The code in the figure and the facts representing the problem are fed to the ASP solver CLINGO, which either returns a (stable) model for the program or reports that no such model exists. In the former case, a policy that solves P_c is obtained from the atoms `pi(S,A)` made true by the model.

The set of ground atoms providing a flat representation of the problem P_c contains the atoms `STATE(s)`, `ACTION(a)`, and `TRANSITION(s,a,s')` for each (reachable) state s, ground action a and transition $s' \in F(a, s)$ found in a reachability analysis from the initial state s_0. In addition, the set includes the atoms `INITIAL(s0)`, `GOAL(s)` for goal states s, and `ASET(i,a)` and `BSET(i,b)` for a fairness assumption A_i/B_i in C if $a \in A_i$ and $b \in B_i$ respectively.

The program for the FOND$^+$ problem P_c is denoted as $T(P_c)$, while $T(P_c, \pi)$ is used to refer to the program $T(P_c)$ but with the line 2 in Figure 2 replaced by facts `pi(s,a)` when $\pi(s) = a$ for a given policy π, and the integrity constraint in line 23 removed. A model M for $T(P_c)$ encodes a policy π_M where $\pi_M(s) = a$ iff `pi(s,a)` holds in M. The formal properties of the FOND-ASP planner are as follows:

Theorem 21. *Let $P_C = \langle P, C\rangle$ be a FOND$^+$ problem, and let π be a policy for P. Then,*

1. *There is a **unique stable model** M of $T(P_c, \pi)$, and `terminate(s)` $\in M$ iff s terminates (Definition 15).*
2. *The policy π solves P_c iff the model M for $T(P_c, \pi)$ satisfies the integrity constraint in line 23 in Figure 2.*
3. *M is a model of $T(P_c)$ iff M is the model of $T(P_c, \pi_M)$ and M satisfies the integrity constraints. Thus,* FOND-ASP *is a sound and complete planner for FOND$^+$.*
4. *Deciding if $T(P_c, \pi)$ has a model is in P; i.e.,* FOND-ASP *runs in non-deterministic exponential time.*

Complexity

A direct consequence of Theorem 21 is that the plan-existence decision problem for FOND$^+$ is in NEXP (i.e., non-deterministic exponential time). Since FOND problems are easily reduced to FOND$^+$ problems (Theorem 13) and the plan-existence for FOND is EXP-Hard (Littman, Goldsmith, and Mundhenk 1998; Rintanen 2004), plan-existence for FOND$^+$ is EXP-Hard as well. We conjecture that the NEXP bound is loose and that plan-existence for FOND$^+$ is EXP-Complete. In contrast, LTL planning and synthesis is 2EXP-Complete (Pnueli and Rosner 1989).

Theorem 22. *The plan-existence problem for FOND$^+$ problems is in NEXP and it is EXP-Hard.*

Experiments

We tested FOND-ASP on three classes of problems:[2] FOND problems, QNPs, and more expressive FOND$^+$ problems that do not fit in either class and that can only be addressed using LTL engines. On each class, we compare FOND-ASP with the FOND solvers FOND-SAT (Geffner and Geffner 2018) and PRP (Muise, McIlraith, and Beck 2012), the QNP solver QNP2FOND (Bonet and Geffner 2020) using FOND-SAT and PRP as the underlying FOND solver, and the LTL-synthesis tool STRIX (Luttenberger, Meyer, and Sickert 2020). The pure (strong and strong-cyclic) FOND problems are those in the FOND-SAT distribution, the QNPs are those by (Bonet and Geffner 2020) and two new families of instances that grow in size with a parameter. For more expressive FOND$^+$ planning problems, four new families of problems are introduced that extend the new QNPs with fair and adversarial actions, with only some being solvable. The domain and goals of these problems are encoded in LTL in the usual way, while the fairness assumptions A/B are encoded as described in the introduction. In all the experiments, time and memory bounds of 1/2 hour and 8GB are enforced.

The results are detailed below. In summary, we observe the following. For pure FOND benchmarks, FOND-ASP does not compete with specialized planners like PRP or FOND-SAT as these problems span (reachable) state spaces that are just too large. For QNPs, on the other hand, FOND-ASP does better than FOND-SAT but worse than PRP on the FOND translations. For expressive FOND$^+$ problems, where these planners cannot be used at all, FOND-ASP performs much better than STRIX on both solvable and unsolvable problems.

FOND Benchmarks

FOND-ASP managed to solve a tiny fraction of the benchmarks used for strong and strong-cyclic planning in the FOND-SAT distribution. The number of reachable states in these problems is large (tens of thousands or more) and the size of the grounded ASP program is quadratic in that number. In general, this seems to limit the scope of FOND-ASP to problems with no more than one thousand states approximately, as suggested by the results in Table 1. We have observed however that sometimes FOND-ASP manages to solve strong planning problems with more than 100,000 states. This may have to do with CLINGO's grounder or with the state space topology; we do not know the exact reason yet.

QNP Problems

The two families of QNPs involve the numerical variables $\{x_i\}_{i=1}^n$ that have all positive values in the initial state. The goal is to achieve $x_n = 0$. Problems in the QNP1 family are

[2] Planner available at https://github.com/idrave/FOND-ASP

```
1  % policy, edges, and connectedness
2  { pi(S,A) : ACTION(A) } = 1 :- STATE(S), not GOAL(S).
3  edge(S,T) :- pi(S,A), TRANSITION(S,A,T).
4  connected(S,T) :- edge(S,T).
5  connected(S,T) :- connected(S,X), edge(X,T), S != X.
6
7  % blocked(S,T) iff there is no (S,T)-path, or all such paths have a terminating state
8  blocked(S,T) :- STATE(S), STATE(T), not connected(S,T).
9  blocked(S,T) :- connected(S,T), terminate(S).
10 blocked(S,T) :- connected(S,T), terminate(T).
11 blocked(S,T) :- connected(S,T), blocked(X,T) : edge(S,X), connected(X,T).
12
13 % fair(S) iff for some A/B, pi(S) in A and each cycle over S that passes
14 %             over X such that pi(X) in B contains a terminating state
15 fair(S) :- pi(S,A), ASET(I,A), blocked(X,S) : pi(X,B), BSET(I,B), not blocked(S,X).
16
17 % terminating states
18 terminate(S) :- GOAL(S).
19 terminate(S) :- fair(S), edge(S,T), terminate(T).
20 terminate(S) :- not fair(S), edge(S,_), terminate(T) : edge(S,T).
21
22 % reachable states must terminate
23 :- reachable(S), not terminate(S).
24 reachable(S) :- INITIAL(S).
25 reachable(S) :- reachable(X), not GOAL(X), edge(X,S).
```

Figure 2: The concise encoding in CLINGO of the ASP-based FOND$^+$ planner tested (FOND-ASP). The FOND$^+$ problem enters through the predicates `STATE/1`, `ACTION/1`, `INITIAL/1`, `GOAL/1`, `TRANSITION/3`, `ASET/2` and `BSET/2`, where `ASET(i,A)` (resp. `BSET(i,A)`) iff action `A` belongs to A_i (resp. B_i) in the fairness assumption A_i/B_i. In CLINGO; the syntax "`P␣:␣<cond>`" used in lines 11, 15, and 20 stands for the implicitly universally quantified conditional "if `<cond>` then `P`".

solved by means of n sequential simple loops, while problems in the QNP2 family are solved using n nested loops. The actions for problems in QNP1 are $b = \langle \neg p; p \rangle$, $a_1 = \langle p; \neg p, x_1{\downarrow} \rangle$, and $a_i = \langle p, x_{i-1} = 0; \neg p, x_i{\downarrow} \rangle$ for $1 < i \leq n$, while those for QNP2 are $b = \langle \neg p; p \rangle$, $a_1 = \langle p; \neg p, x_1{\downarrow} \rangle$, and $a_i = \langle p, x_{i-1} = 0; \neg p, x_{i-1}{\uparrow}, x_i{\downarrow} \rangle$, $1 < i \leq n$.

Table 1 shows the results for values of n in $\{2, 3, \ldots, 10\}$ and different planners, along with the number of reachable states in each problem. As can be seen, QNP2FOND/PRP is the planner that scales best, followed by FOND-ASP, QNP2FOND/FOND-SAT, and STRIX at the end. As mentioned, the performance of FOND-ASP is harmed by a large number of reachable states. While the number of states for the FOND translation produced by QNP2FOND is much larger, as the translation involves extra propositions, this number does not necessarily affect the performance of FOND planners like FOND-SAT and PRP that can compute compact policies. It is also interesting to see how quickly the performance of the LTL engine STRIX degrades; it cannot even solve qnp1-06 which has 14 states. The table also shows results for QNP problems that capture abstractions for four generalized planning problems, all of which involve small state spaces (Bonet and Geffner 2020).

More Expressive FOND$^+$ Problems

The third class of instances consists of four families of problems obtained from the two QNP families above. The new problems are not "pure" QNPs, as they also involve actions with non-deterministic effects on boolean variables that can be adversarial or fair. Thus, these problems cannot be translated into FOND problems for the use of planners such as PRP or FOND-SAT. For each family QNP1 and QNP2, two new families $f01$ and $f11$ of problems are obtained by replacing the action $b = \langle \neg p; p \rangle$ by the non-deterministic action $b' = \langle \neg p; \textit{oneof}\ \{p, \neg p\} \rangle$, leaving the actions a_i untouched. Since the action b' does not appear in any fairness assumption, it is adversarial and thus no problem in the class $f01$ has a solution as the "adversary" may always choose to leave p false. The family $f11$ is obtained on top of $f01$ by adding two additional booleans q and r, and two actions $c = \langle \neg q; r, \textit{oneof}\ \{q, \neg q\} \rangle$ and $d = \langle r; q, \neg r \rangle$ such that: 1) the actions a_i are modified by adding q as precondition and $\neg q$ as effect, and 2) the fairness assumption A/B with $A = \{b'\}$ and B empty is added. The problems in $f11$ thus involve the QNP-like actions a_i, the fair action b', and the adversarial action c, and they all have a solution.

Table 2 shows the result for FOND-ASP and STRIX as these are the only solvers able to handle the combination of fairness assumptions. As it can be seen, FOND-ASP scales better than STRIX on all of these problems, the solvable ones (families $f11$) and the unsolvable ones (families $f01$).

We finally tested FOND-ASP over the seven problems considered in a recent approach to program synthesis over unbounded data structures (Bonet et al. 2020). Although the original specifications are in LTL, these can be all expressed in FOND$^+$ using different types of fairness assumptions. The problems are solved easily by both FOND-ASP and STRIX as their reachable state spaces have very few states.

problem	#states	QNP2FOND		STRIX	FOND-ASP
		FOND-SAT	PRP		
qnp1-02	6	0.08	0.09	3.35	0.00
qnp1-03	8	0.13	0.11	2.63	0.00
qnp1-04	10	0.28	0.14	5.21	0.00
qnp1-05	12	0.60	0.14	98.34	0.01
qnp1-06	14	1.27	0.15	—	0.01
qnp1-07	16	2.54	0.15	—	0.01
qnp1-08	18	5.54	0.17	—	0.01
qnp1-09	20	12.96	0.21	—	0.02
qnp1-10	22	26.70	0.19	—	0.02
qnp2-02	8	0.20	0.18	2.33	0.00
qnp2-03	16	1.77	0.30	2.31	0.01
qnp2-04	32	10.00	0.58	14.25	0.04
qnp2-05	64	50.24	1.15	885.37	0.20
qnp2-06	128	302.80	2.53	—	1.26
qnp2-07	256	1,969.35	4.02	—	7.14
qnp2-08	512	—	6.96	—	54.37
qnp2-09	1,024	—	13.22	—	***
qnp2-10	2,048	—	21.94	—	***
Clear	4	0.23	0.13	1.53	0.00
On	16	3.69	0.20	3.01	0.01
Delivery	12	4.07	0.27	1.50	0.01
Gripper	12	15.43	1.61	2.47	0.02

Table 1: Results for three families of QNPs for QNP2FOND paired with the FOND solvers FOND-SAT and PRP, STRIX (QNP translated to LTL), and FOND-ASP. Entries '—' and '***' denote out of time and memory, resp. Time is in secs.

problem	$f01$ (unsolvable)			$f11$ (solvable)		
	#states	STRIX	FOND-ASP	#states	STRIX	FOND-ASP
qnp1-$f$xx-02	6	3.44	0.00	24	6.07	0.03
qnp1-$f$xx-03	8	2.42	0.01	32	6.20	0.04
qnp1-$f$xx-04	10	4.13	0.01	40	98.58	0.07
qnp1-$f$xx-05	12	93.85	0.01	48	—	0.11
qnp1-$f$xx-06	14	—	0.01	56	—	0.14
qnp1-$f$xx-07	16	—	0.01	64	—	0.19
qnp1-$f$xx-08	18	—	0.01	72	—	0.25
qnp1-$f$xx-09	20	—	0.02	80	—	0.39
qnp1-$f$xx-10	22	—	0.02	88	—	0.34
qnp2-$f$xx-02	8	3.22	0.00	32	5.85	0.04
qnp2-$f$xx-03	16	2.25	0.01	64	8.16	0.21
qnp2-$f$xx-04	32	11.38	0.04	128	236.89	1.55
qnp2-$f$xx-05	64	873.09	0.21	256	—	15.45
qnp2-$f$xx-06	128	—	1.25	512	—	46.67
qnp2-$f$xx-07	256	—	12.13	1,024	—	***
qnp2-$f$xx-08	512	—	39.56	2,048	—	***
qnp2-$f$xx-09	1,024	—	***	4,096	—	***
qnp2-$f$xx-10	2,048	—	***	8,192	—	***

Table 2: Results for four families of solvable/unsolvable FOND$^+$ problems obtained from the QNPs in Table 1 by playing with the fairness assumptions. These problems are handled only by STRIX and FOND-ASP. Entries '—' and '***' denote out of time and memory, resp. Time is in secs.

Related Work

The work is related to three threads: SAT-based FOND planning, QNPs, and LTL synthesis. The SAT-based FOND planner by Chatterjee, Chmelík, and Davies (2016) expands the state space in full, like FOND-ASP, but a more recent version computes compact policies and provides support for Dual FOND planning (Geffner and Geffner 2018). We have used answer set programs as opposed to CNF encodings exploiting their high-level modeling language, the natural support for inductive definitions, and the competitive performance of CLINGO (Gebser et al. 2019). FOND-ASP is also a novel QNP planner which can handle non-deterministic effects on boolean variables. The formulation actually brings QNP planning into the realm of standard FOND planning by dealing with the underlying fairness assumptions explicitly.

The use of fairness assumptions connects also to works on LTL planning and synthesis (Camacho, Bienvenu, and McIlraith 2019; Aminof et al. 2019), and to works addressing temporally extended goals (De Giacomo and Vardi 1999; Patrizi, Lipovetzky, and Geffner 2013; Camacho et al. 2017; Camacho, Bienvenu, and McIlraith 2019; Aminof, De Giacomo, and Rubin 2020). Our work can be seen as a special case of planning under LTL assumptions (Aminof et al. 2019) that targets an LTL fragment that is relevant for FOND planning and is computationally simpler. While it is possible to express FOND$^+$ tasks as LTL syntheses problems, and we have shown how to do that, it remains to be seen whether the task can be expressed in a *restricted* LTL fragment that admits more efficient techniques. While the *strong* fairness assumption on action effects that is required cannot be *directly* encoded in GR(1) (Bloem et al. 2012), strong-cyclic FOND planning has been encoded in Büchi Games (D'Ippolito, Rodríguez, and Sardiña 2018), a special case of GR(1). It remains to be investigated whether that encoding can be extended to deal with *conditional* fairness.

Summary

We have formulated an extension of FOND planning that makes use of explicit fairness assumptions of the form A/B where A and B are disjoints sets of actions. While in Dual FOND planning actions are labeled as fair or unfair, in FOND$^+$ planning these labels are a function of the trajectories and the fairness assumptions: an action $a \in A$ is deemed fair in a recurrent state if a suitable condition on B holds. In this way, FOND$^+$ generalizes strong, strong-cyclic, Dual FOND planning, and also QNP planning, which is actually the only planning setting, excluding LTL planning, that makes use of the conditions B. We have implemented an effective FOND$^+$ planner by reducing the problem to answer set programs using CLINGO, and evaluated its performance in relation to FOND and QNP planners, which handle less expressive problems, and LTL synthesis tools, which handle more expressive ones. We have shown that FOND$^+$ is in NEXP but have not shown yet whether it is in EXP, like FOND and QNP planning.

Acknowledgments

We thank the CLINGO team for the tool. The work is partially supported by an ERC Advanced Grant (No 885107), the project TAILOR, funded by an EU Horizon 2020 Grant (No 952215), and by the Knut and Alice Wallenberg (KAW) Foundation under the WASP program. Hector Geffner is a Wallenberg Guest Professor at Linköping University, Sweden.

References

Aminof, B.; De Giacomo, G.; Murano, A.; and Rubin, S. 2019. Planning under LTL Environment Specifications. In *Proc. ICAPS*, 31–39.

Aminof, B.; De Giacomo, G.; and Rubin, S. 2020. Stochastic Fairness and Language-Theoretic Fairness in Planning in Nondeterministic Domains. In *Proc. ICAPS*, 20–28.

Bertsekas, D.; and Tsitsiklis, J. 1996. *Neuro-Dynamic Programming*. Athena Scientific.

Bloem, R.; Jobstmann, B.; Piterman, N.; Pnueli, A.; and Sa'ar, Y. 2012. Synthesis of Reactive(1) designs. *Journal of Computer and System Sciences* 78(3): 911–938.

Bonet, B.; De Giacomo, G.; Geffner, H.; Patrizi, F.; and Rubin, S. 2020. High-Level Programming via Generalized Planning and LTL Synthesis. In *Proc. KR*, 152–161.

Bonet, B.; De Giacomo, G.; Geffner, H.; and Rubin, S. 2017. Generalized Planning: Non-Deterministic Abstractions and Trajectory Constraints. In *Proc. IJCAI*, 873–879.

Bonet, B.; Frances, G.; and Geffner, H. 2019. Learning features and abstract actions for computing generalized plans. In *Proc. AAAI*, 2703–2710.

Bonet, B.; and Geffner, H. 2020. Qualitative Numeric Planning: Reductions and Complexity. *JAIR* 69: 923–961.

Brewka, G.; Eiter, T.; and Truszczyński, M. 2011. Answer set programming at a glance. *Comm. ACM* 54(12): 92–103.

Calvanese, D.; De Giacomo, G.; and Vardi, M. Y. 2002. Reasoning about actions and planning in LTL action theories. *KR* 2: 593–602.

Camacho, A.; Bienvenu, M.; and McIlraith, S. A. 2019. Towards a Unified View of AI Planning and Reactive Synthesis. In *Proc. AAAI*, 58–67.

Camacho, A.; and McIlraith, S. A. 2016. Strong-Cyclic Planning when Fairness is Not a Valid Assumption. In *Proc. KnowProS Workshop*.

Camacho, A.; Triantafillou, E.; Muise, C. J.; Baier, J. A.; and McIlraith, S. A. 2017. Non-Deterministic Planning with Temporally Extended Goals: LTL over Finite and Infinite Traces. In *Proc. AAAI*.

Chatterjee, K.; Chmelík, M.; and Davies, J. 2016. A symbolic SAT-based algorithm for almost-sure reachability with small strategies in POMDPs. In *Proc. AAAI*, 3225–3232.

Cimatti, A.; Pistore, M.; Roveri, M.; and Traverso, P. 2003. Weak, strong, and strong cyclic planning via symbolic model checking. *Artificial Intelligence* 147(1-2): 35–84.

Cimatti, A.; Roveri, M.; and Traverso, P. 1998a. Automatic OBDD-based generation of universal plans in non-deterministic domains. In *Proc. AAAI*, 875–881.

Cimatti, A.; Roveri, M.; and Traverso, P. 1998b. Strong Planning in Non-Deterministic Domains Via Model Checking. In *Proc. AIPS*, 36–43.

Ciolek, D.; D'Ippolito, N.; Pozanco, A.; and Sardiña, S. 2020. Multi-Tier Automated Planning for Adaptive Behavior. In *Proc. ICAPS*, volume 30, 66–74.

Daniele, M.; Traverso, P.; and Vardi, M. Y. 1999. Strong Cyclic Planning Revisited. In *Recent Advances in AI Planning*, 35–48. Springer.

De Giacomo, G.; and Vardi, M. Y. 1999. Automata-Theoretic Approach to Planning for Temporally Extended Goals. In *Proc. ECP*, volume 1809 of *LNCS*, 226–238.

D'Ippolito, N.; Rodríguez, N.; and Sardiña, S. 2018. Fully Observable Non-deterministic Planning as Assumption-Based Reactive Synthesis. *JAIR* 61: 593–621.

Gebser, M.; Kaminski, R.; Kaufmann, B.; and Schaub, T. 2012. Answer set solving in practice. *Synthesis lectures on artificial intelligence and machine learning* 6(3): 1–238.

Gebser, M.; Kaminski, R.; Kaufmann, B.; and Schaub, T. 2019. Multi-shot ASP solving with clingo. *Theory and Practice of Logic Programming* 19(1): 27–82.

Geffner, H.; and Bonet, B. 2013. *A Concise Intro. to Models and Methods for Automated Planning*. Morgan-Claypool.

Geffner, T.; and Geffner, H. 2018. Compact policies for Non-deterministic Fully Observable Planning as SAT. In *Proc. ICAPS*, 88–96.

Helmert, M. 2002. Decidability and Undecidability Results for Planning with Numerical State Variables. In *Proc. AIPS*.

Hu, Y.; and De Giacomo, G. 2011. Generalized planning: Synthesizing plans that work for multiple environments. In *Proc. IJCAI*, 918–923.

Lifschitz, V. 2019. *Answer set programming*. Springer.

Littman, M. L.; Goldsmith, J.; and Mundhenk, M. 1998. The computational complexity of probabilistic planning. *JAIR* 9: 1–36.

Luttenberger, M.; Meyer, P. J.; and Sickert, S. 2020. Practical synthesis of reactive systems from LTL specifications via parity games. *Acta Informatica* 57(1-2): 3–36.

Mattmüller, R.; Ortlieb, M.; Helmert, M.; and Bercher, P. 2010. Pattern Database Heuristics for Fully Observable Nondeterministic Planning. In *Proc. ICAPS*, 105–112.

Muise, C.; McIlraith, S. A.; and Beck, J. C. 2012. Improved Non-deterministic Planning by Exploiting State Relevance. In *Proc. ICAPS*, 172–180.

Patrizi, F.; Lipovetzky, N.; and Geffner, H. 2013. Fair LTL Synthesis for Non-Determinsistic Systems using Strong Cyclic Planners. In *Proc. IJCAI*.

Pnueli, A.; and Rosner, R. 1989. On the synthesis of an asynchronous reactive module. In *ICALP*, 652–671.

Rintanen, J. 2004. Complexity of Planning with Partial Observability. In *Proc. ICAPS*, 345–354.

Rodriguez, I. D.; Bonet, B.; Sardiña, S.; and Geffner, H. 2021. Flexible FOND Planning with Explicit Fairness Assumptions. *arXiv preprint:2103.08391* .

Srivastava, S.; Immerman, N.; and Zilberstein, S. 2011. A new representation and associated algorithms for generalized planning. *Artificial Intelligence* 175(2): 615–647.

Srivastava, S.; Zilberstein, S.; Immerman, N.; and Geffner, H. 2011. Qualitative Numeric Planning. In *Proc. AAAI*.