Automata-Theoretic Foundations of FOND Planning for LTL_f and LDL_f Goals

Giuseppe De Giacomo
Univ. Roma "La Sapienza", Italy
degiacomo@dis.uniroma1.it

Sasha Rubin
Univ. Napoli "Federico II", Italy
sasha.rubin@unina.it

Abstract

We study planning for LTL_f and LDL_f temporally extended goals in nondeterministic fully observable domains (FOND). We consider both strong and strong cyclic plans, and develop foundational automata-based techniques to deal with both cases. Using these techniques we provide the computational characterization of both problems, separating the complexity in the size of the domain specification from that in the size of the formula. Specifically we establish them to be EXPTIME-complete and 2EXPTIME-complete, respectively, for both problems. In doing so, we also show 2EXPTIME-hardness for strong cyclic plans, which was open.

1 Introduction

We study planning in fully observable nondeterministic domains (FOND) [Ghallab *et al.*, 2004; Geffner and Bonet, 2013] for temporally extended goals expressed in LTL_f and its extension LDL_f. We consider two variants of planning: in FOND_{unr}, plans are guaranteed to produce traces that satisfy the goal; in FOND_{fair}, this is guaranteed only in *fair* environments [Daniele *et al.*, 1999; Pistore and Traverso, 2001; Cimatti *et al.*, 2003; Sardiña and D'Ippolito, 2015].

Linear temporal logic on finite traces (LTL_f*)* has been used extensively in AI [Bacchus and Kabanza, 2000; Baier and McIlraith, 2006; De Giacomo and Vardi, 2013], as well as in other areas of computer science such as in business process modeling [van der Aalst *et al.*, 2009]. In Planning, notably, LTL_f can be used for expressing trajectory constraints in PDDL 3.0 [Gerevini *et al.*, 2009].

Linear dynamic logic on finite traces (LDL_f*)* [De Giacomo and Vardi, 2013] is a proper extension of LTL_f by regular expressions. It can capture procedural constraints on executions, as in [Fritz and McIlraith, 2007; Baier *et al.*, 2008], which are typically expressed as regular expressions that must be fulfilled by traces. The logics $\mathrm{LTL}_f/\mathrm{LDL}_f$ are also used to express non-Markovian rewards/goals in extensions of MDPs [Camacho *et al.*, 2017a; Brafman *et al.*, 2018].

In this paper we make a careful study of the complexity of solving FOND_{unr} and FOND_{fair}. We observe that from results in literature, reviewed below, it is easy to see that both of these are in 2EXPTIME. However, such results *blur the distinction* between the *domain* specification (typically large) and the *goal* (typically small). In our analysis we want to distinguish these two source of complexity, i.e., the *domain complexity* and the *goal complexity*.

FOND Planning. One way to see that FOND_{unr} planning is solvable in 2EXPTIME is to reduce it to reactive synthesis. *Reactive Synthesis* is a problem that is deeply related to planning in fully observable nondeterministic domains. It can be seen as a generalization of planning in non-deterministic domains, where both the goal and the domain are represented as formulas. Reactive synthesis for $\mathrm{LTL}_f/\mathrm{LDL}_f$ formulas is 2EXPTIME-complete [De Giacomo and Vardi, 2015].

For the *upper bound*, observe that FOND_{unr} with $\mathrm{LTL}_f/\mathrm{LDL}_f$ goals is polynomially reducible to reactive synthesis for $\mathrm{LTL}_f/\mathrm{LDL}_f$ goals. Indeed, we can polynomially represent the domain specification in terms of LTL_f formulas expressing preconditions and effects. This gives the 2EXPTIME upper bound. However, this reduction does not allow one to study the domain/goal complexities since it compiles both the domain and the goal into a single formula.

For the *lower bound*, observe that $\mathrm{LTL}_f/\mathrm{LDL}_f$ reactive synthesis is polynomially reducible to FOND_{unr} planning in a very simple domain that allows all actions and has all possible nondeterministic effects. This establishes that FOND_{unr} with $\mathrm{LTL}_f/\mathrm{LDL}_f$ goals is also 2EXPTIME-hard.

FOND planning in fair environments. Turning to the 2EXPTIME *upper bound* for FOND_{fair}, we note that fairness is a property that requires the analysis of traces that are infinite. Hence we cannot reduce FOND_{fair} directly to $\mathrm{LTL}_f/\mathrm{LDL}_f$ synthesis. That said, we can indeed reduce it to LTL/LDL synthesis, i.e., synthesis on infinite traces, which is 2EXPTIME-complete [Pnueli and Rosner, 1989].[1] However, synthesis still appears to be prohibitive in the infinite-trace setting, not so much for the 2EXPTIME-completeness, but for the difficulties of finding good algorithms for automata determinization, a crucial step in the solution, see, e.g., [Fogarty *et al.*, 2013]. Fortunately, efficient techniques for solving FOND_{fair} for $\mathrm{LTL}_f/\mathrm{LDL}_f$ goals have been proposed in [Camacho *et al.*, 2017b], where a sound and complete translation into an expo-

[1] We remark that LDL on infinite traces [Vardi, 2011] can be seen as a clean version of standard formalisms used in verification such as Intel *ForSpec* [Armoni et al., 2002] and the standard *PSL* [Eisner and Fisman, 2006]. The synthesis techniques developed in [Pnueli and Rosner, 1989] immediately extends to LDL on infinite traces.

nentially larger FOND*fair* problem for the standard reachability goal has been presented, implemented and experimented in practice. This compilation not only gives the 2EXPTIME upper bound, but better algorithms. However, it still blurs the complexity coming from the domain and that coming from the LTL$_f$/LDL$_f$ goal since it compiles both the domain and the LTL$_f$/LDL$_f$ goal into the domain. The existence of matching *lower bounds* has been open until now.[2]

Our contribution. We study the complexity of planning for LTL$_f$/LDL$_f$ goals separating the analysis wrt the domain and the goal. For the *upper bounds*, we develop simple, direct and elegant automata-based techniques to deal both with FOND*unr* and FOND*fair* for LTL$_f$/LDL$_f$ goals. Using these techniques we separate the domain and goals as sources of complexity, and characterize both problems as EXPTIME in the size of domain specification (expressed compactly, e.g., as in PDDL) and as 2EXPTIME in the size of the LTL$_f$/LDL$_f$ goal. For the *lower bounds*, we observe that EXPTIME-hardness in the size of the domain already holds for reachability goals [Rintanen, 2004]. For FOND*unr*, 2EXPTIME-hardness in the size of the goal comes from the mentioned reduction from LTL$_f$ synthesis, and here we extend this reduction to show 2EXPTIME-hardness also for FOND*fair*. We actually strengthen this characterization by showing that the smallest plan/strategy fulfilling an LTL$_f$ goal may indeed need memory that is doubly-exponential in the size of the goal. This result applies to FOND*fair* as well as to FOND*unr*. Finally, we develop a tight correspondence between solving FOND*fair* with LTL$_f$/LDL$_f$ goals and solving probabilistic planning problems with LTL$_f$/LDL$_f$ goals, analogous to the well-known one for reachability goals [Rintanen, 2004].

2 Automata, LTL$_f$ and LDL$_f$

Linear-time Temporal Logic over finite traces (LTL$_f$) has the same syntax as standard LTL, but is interpreted over finite traces (instead of infinite traces). For instance, the formula $\Diamond G$ where G is a Boolean formula expresses that eventually G holds. Linear Dynamic Logic (LDL$_f$) is a proper extension of LTL$_f$ that is able to capture full regular expressions over traces. We refer to [De Giacomo and Vardi, 2013] for more details. Here, we focus on the property that: for every LTL$_f$/LDL$_f$ formula φ there is a nondeterministic finite word automata (NFA) that accepts the traces τ that satisfy φ, and the size of the NFA is at most exponential in the size of the formula [De Giacomo and Vardi, 2013]. Such an NFA, as any NFA, can be transformed, e.g., by the well-known subset construction [Rabin and Scott, 1959], into a deterministic finite state automaton (DFA), which is in general exponential in the NFA, though in many cases the resulting DFA is comparable to, if not smaller (after minimization) than the original NFA [Tabakov and Vardi, 2005].

[2]An alternative way of studying the complexities separately is to reduce to model checking variants of ATL* [Alur *et al.*, 2002]. However, we would get into the same difficulties as for LTL synthesis mentioned above. Hence, here we follow a different path.

3 FOND planning with LTL$_f$/LDL$_f$ goals

Following [Geffner and Bonet, 2013], a *nondeterministic domain* is a tuple $\mathcal{D} = (2^{\mathcal{F}}, \mathcal{A}, s_0, \delta, \alpha)$ where: $\mathcal{F}$ is a set of *fluents* (atomic propositions); $\mathcal{A}$ is a set of *actions* (atomic symbols); $2^{\mathcal{F}}$ is the set of states; s_0 is the initial state (initial assignment to fluents); $\alpha(s) \subseteq \mathcal{A}$ represents *action preconditions*; $(s, a, s') \in \delta$ with $a \in \alpha(s)$ represents *action effects (including frame assumptions)*. Such a domain is assumed to be represented *compactly* (e.g. in PDDL), hence we *consider the size of the domain as the cardinality of* $\mathcal{F}$, i.e., logarithmic in the number of states.

Intuitively, a nondeterministic domain evolves as follows: from a given state s, the agent chooses an action $a \in \alpha(s)$, after which the environment chooses a successor state s' with $(s, a, s') \in \delta$. The agent can choose its action based on the history of states so far (i.e., the agent has full observation).

We now define what it means to solve a planning problem on $\mathcal{D}$. A *trace* of $\mathcal{D}$ is a finite or infinite sequence $s_0, a_0, s_1, a_1, \cdots$ where s_0 is the initial state, and $a_i \in \alpha(s_i)$ and $s_{i+1} = \delta(s_i, a_i)$ for each s_i, a_i in the trace. A *strategy* (or *plan*) is a partial function $f : (2^{\mathcal{F}})^+ \to \mathcal{A}$ such that for every $u \in (2^{\mathcal{F}})^+$, if $f(u)$ is defined then $f(u) \in \alpha(last(u))$, i.e., it selects applicable actions. If $f(u)$ is undefined, we write $f(u) = \bot$. A trace τ is *generated by* f, or simply an f*-trace*, if (i) if $s_0, a_0, \cdots s_i, a_i$ is a prefix of τ then $f(s_0 s_1 \cdots s_i) = a_i$, and (ii) if τ is finite, say $\tau = s_0, a_0, \ldots, a_{n-1}, s_n$, then $f(s_0 s_1 \cdots s_n) = \bot$.

Given a domain $\mathcal{D}$ and a LTL$_f$/LDL$_f$ *goal* formula φ over atoms $\mathcal{F} \cup \mathcal{A}$, a strategy f is a *strong solution to* $\mathcal{D}$ *for goal* φ if every f-trace of $\mathcal{D}$ is finite and satisfies φ. The FOND*unr* problem takes as input $\mathcal{D}$ and φ, and decides if there exists a strong solution to $\mathcal{D}$ for goal φ. In case such a strategy exists, we are also interested in computing one.

Relationship with reachability goals. In the classical setting the goal has the form $\Diamond G$, where G is a propositional formula over atoms $\mathcal{F}$ (i.e., the objective is to reach a domain state satisfying G). This form of planning has been thoroughly studied in literature under the name *conditional planning with full observability*, e.g., [Rintanen, 2004]. Typically such plans are represented as conditional trees. Alternatively we can compute a *universal plan* [Schoppers, 1987], i.e., a *memoryless policy* that given the current state returns the action to do. The two kind solutions have the same computational cost, although algorithms for universal plans are typically less efficient. Note that for reachability goals $\Diamond G$ it is guaranteed that if a strong solution exists there is one which is memoryless (i.e., a memoryless policy). This is not true for general LTL$_f$/LDL$_f$ goals, which is why we consider plans to be (memoryfull) strategies. Solving FOND*unr* for goals of the form $\Diamond G$ is known to be EXPTIME-complete in the size of the domain [Rintanen, 2004].

Automata-theoretic solution. Next, we use automata-theoretic techniques to solve FOND*unr* planning for LTL$_f$/LDL$_f$ goals, Algorithm 1 (for simplicity, complexities in parenthesis are wrt explicit representation of domain $\mathcal{D}$, i.e., in the number of states).

Algorithm 1: **FOND*unr* for LDL$_f$/LTL$_f$ goals**
Given LTL$_f$/LDL$_f$ domain $\mathcal{D}$ and goal φ

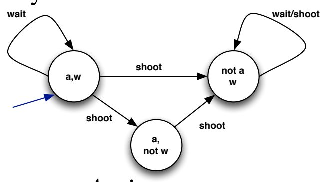
```
1: Compute DFA corresponding to $\mathcal{D}$ (poly)
2: Compute NFA for $\varphi$ (exp)
3: Determinize NFA to DFA (exp)
4: Compute product with DFA of $\mathcal{D}$ (poly)
5: Solve DFA game (poly)
6: Return winning strategy if it exists
```

The various steps are described in detail below.

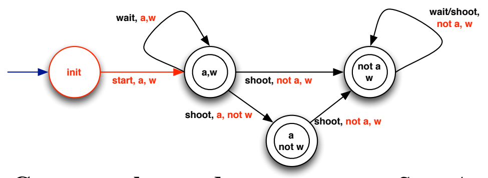
Nondeterministic domains as automata. In Step 1 we transform the nondeterministic domain $\mathcal{D} = (2^{\mathcal{F}}, \mathcal{A}, s_0, \delta, \alpha)$ into an automaton A_D recognizing all its traces, defined as $A_{\mathcal{D}} = (2^{\mathcal{F}\cup\mathcal{A}}, (2^{\mathcal{F}} \cup \{s_{init}\}), s_{init}, \varrho, F)$ with: alphabet $2^{\mathcal{F}\cup\mathcal{A}}$ (actions $\mathcal{A}$ include dummy *start* action); states $2^{\mathcal{F}} \cup \{s_{init}\}$; initial dummy state s_{init}; final states $F = 2^{\mathcal{F}}$ (all states of the domain are final); transitions $\varrho(s, [a, s']) = s'$ with $a \in \alpha(s)$ and $(s, a, s') \in \delta$, and $\varrho(s_{init}, [start, s_0]) = s_0$. For convenience we use the notation $[a, s']$ to stand for $\{a\} \cup s'$. Note that in the automaton $A_{\mathcal{D}}$ we consider actions as further fluents in the alphabet.

A critical observation is that A_D *is a* DFA. This is because in order to progress, the automaton *reads both the action and its effect*. In particular, the nondeterminism in the environment (which is devilish) is not translated into a nondeterminism in the automaton (which is angelic).

Example 1 Consider the following simplified version of the classical Yale shooting domain [Hanks and McDermott, 1986], where we have that the turkey is either alive or not and the actions are either shoot and wait with the obvious effects, though with a gun that may be faulty. In particular, shooting with a (supposedly) working gun may either result in killing the turkey or in the turkey remaining alive and the discovery that the gun is not working properly. However shooting (with care) with a gun that does not work properly makes it work and kills the turkey. The domain $\mathcal{D}$ is as follows:

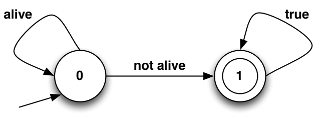


Its corresponding DFA $A_{\mathcal{D}}$ is:



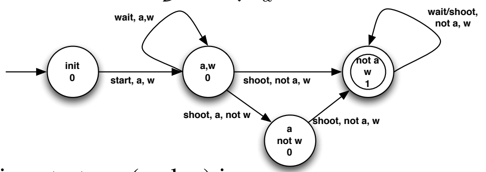
DFA Game on the product automaton Step 4 of the algorithm forms the DFA $A_{\mathcal{D}} \times A_{\varphi}$ for the product of $\mathcal{A}_{\mathcal{D}}$ and $\mathcal{A}_{\varphi}$ (the final states are the product of the final states). Traces in the product come from $\mathcal{D}$, while the final states are determined by A_{φ}. To solve the FOND$_{unr}$ we solve a certain game (Step 5) on the DFA $A_{\mathcal{D}} \times A_{\varphi}$ where the objective of the player is to reach a final state of $A_{\mathcal{D}} \times A_{\varphi}$, i.e., a state that deems the trace from $\mathcal{D}$ as satisfying the LTL$_f$/LDL$_f$ formula φ. Such a game is called a DFA game. Solving such games amounts to finding an agent strategy that is winning, i.e., ensures the reachability objective no matter how the environment behaves. Before looking at DFA games, we consider a couple of examples.

Example 2 Continuing our example, consider a standard reachability goal $\Diamond\neg a$. A corresponding DFA $A_{\Diamond\neg a}$ is as follows (actually any reachability goal $\Diamond G$ with G propositional has an analogous structure):



$A_{\Diamond\neg a}$

The product DFA is: $A_{\mathcal{D}} \times A_{\Diamond\neg a}$ is:



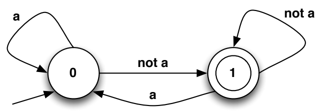
A winning strategy (a plan) is:

$init, 0$	$\rightarrow$	$start$	$a, \neg w, 0$	$\rightarrow$	$shoot$
$a, w, 0$	$\rightarrow$	$shoot$	$\neg a, w, 1$	$\rightarrow$	$\bot$

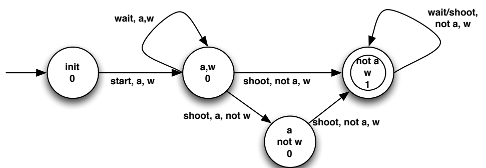
■

Notice that as this example shows $A_{\mathcal{D}}$ and $A_{\mathcal{D}} \times A_{\Diamond G}$ are identical except for the final states, which for $A_{\mathcal{D}} \times A_{\Diamond G}$ are only those where G holds. In other words the states are functionally dependent on those of $\mathcal{D}$ and we do not need additional memory to store them explicitly. This is not the case for general LTL$_f$/LDL$_f$ goals.

Example 3 For the goal $\Diamond\Box\neg a$ a corresponding DFA is:



while the product $A_{\mathcal{D}} \times A_{\Diamond\Box\neg a}$ is



A winning strategy is:

$init, 0$	$\rightarrow$	$start$	$a, \neg w, 0$	$\rightarrow$	$shoot$
$a, w, 0$	$\rightarrow$	$shoot$	$\neg a, w, 1$	$\rightarrow$	$\bot$

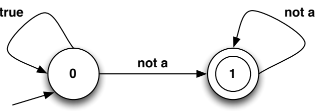
■

One question arises: do we need to determinize the automaton for the formula in the above construction?[3] In fact we cannot use NFA's directly, because of a *basic mismatch*:

– NFA (being based on angelic nondeterminism) have perfect *foresight*, or *clairvoyance*, while
– strategies must be runnable: *depend only on the past*, not the future, and hence can handle only devilish nondeterminism.

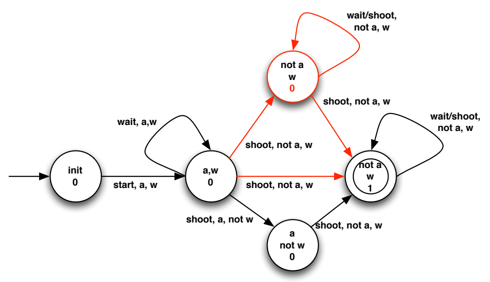
To see the problem, let's look again at our example.

Example 4 Let us consider again $\Diamond\Box\neg a$. An NFA for $A_{\Diamond\Box\neg a}$ is:



[3]After all, for *deterministic domains* we don't need to determinize in order to check for the existence of a domain trace that satisfies an LTL$_f$/LDL$_f$ goal [De Giacomo and Vardi, 2013].

The product $A_{\mathcal{D}} \times A_{\Diamond\Box\neg a}$ is:



The transition (shoot, not alive, working) may change or not the NFA $A_{\Diamond\Box\neg a}$ state to final. If we give this choice to the *environment* we give it *too much power* since it may force looping in a non-final state forever without realizing that we have fulfilled our goal. If we give the choice to the *agent* then we are giving it *clairvoyance*, which is unrealistic. ∎

So, before applying the product we need to determinize the NFA for the LTL$_f$/LDL$_f$ formula. Unfortunately the resulting DFA can be exponential in the NFA, though is not in many cases (we briefly return to this point in the conclusion).

DFA Games. DFA games have been used for LTL$_f$/LDL$_f$ synthesis [De Giacomo and Vardi, 2015], and seamlessly apply to our case. For disjoint sets $\mathcal{F}$ and $\mathcal{A}$, a DFA *game* $\mathcal{G} = (2^{\mathcal{F}\cup\mathcal{A}}, S, s_{init}, \varrho, F)$ has: alphabet $2^{\mathcal{F}\cup\mathcal{A}}$; states S; initial state s_{init}; partial transition function $\varrho : S \times 2^{\mathcal{F}\cup\mathcal{A}} \to S$; and final states F. The game evolves as follows: from a current state s, the agent chooses an action $a \in \mathcal{A}$ (for which $\varrho(s, [a, E])$ is defined for some E), then the environment sets the fluent values E, and then the game moves to the state $\varrho(s, [a, E])$. The evolution ends, and is won by the agent, if a state of F is reached. A *strategy* in $\mathcal{G}$ is a partial function $f : (2^{\mathcal{F}})^* \to \mathcal{A}$. We recast the notions of trace and f-trace to DFA games. A strategy is *winning* if each f-trace in $\mathcal{G}$ starting in the initial state is finite and terminates in F.

To compute winning strategies in DFA games we define the *universal preimage* for $a \in \mathcal{A}, X \subseteq S$: $PreA(a, X) = \{s \in S \mid \forall E \in 2^{\mathcal{F}}, \text{ if } s' = \varrho(s, [a, E]) \text{ then } s' \in X\}$. A state s is in $PreA(a, X)$ if from s and action a, all possible effects E chosen by the environment result in a state of X. Let $PreC(X) = \{s \in S : \exists a \in \mathcal{A}, s \in PreA(a, X)\}$. The winning region Win of a DFA game $\mathcal{G}$, i.e., states from which the agent has a winning strategy, is characterized by *least-fixpoint*, $Win = \mu X.F \cup PreC(X)$, which can be computed by the following algorithm:

- $Win_1 = F$ (the final states of $\mathcal{G}$);
- $Win_{i+1} = Win_i \cup PreC(Win_i)$;
- $Win = \bigcup_{i \leq |S|} Win_i$.

Note: computing Win is *linear* in the number of states of $\mathcal{G}$.

If the initial state s_{init} is in Win, we can extract a winning strategy as follows: define $\omega : Win \setminus F \to 2^{\mathcal{A}}$ as mapping $s \in Win_{i+1} \setminus Win_i$ to the set of all $a \in A$ such that $s \in PreA(a, Win_i)$. *Every way* of restricting $\omega(s)$ to return only one action (chosen arbitrarily) gives a *winning strategy* for $\mathcal{G}$.

Correctness. To see that Algorithm 1 is correct, note that there is a tight correspondence between strategies in $\mathcal{D}$ and strategies in $A_{\mathcal{D}} \times A_{\varphi}$. It is straightforward to see that a strong solution to $\mathcal{D}$ corresponds to a winning strategy in the DFA Game $A_{\mathcal{D}} \times A_{\varphi}$, and vice versa.

Notice that the algorithm for solving DFA games generates memoryless strategies for the DFA game, which include in the state the status of the formula. These strategies correspond to memoryfull strategies in the original domain $\mathcal{D}$ (the memory being the states of the DFA for the goal).

Complexity analysis. For the upper bounds, generate the states of $\mathcal{D}$ explicitly and apply Alg. 1, and for the lower bounds consider hardness of standard FOND$_{unr}$ [Rintanen, 2004] and LTL$_f$ synthesis [De Giacomo and Vardi, 2015]:

Theorem 1 *Solving* FOND$_{unr}$ *for* LTL$_f$/LDL$_f$ *goals is:*

- *EXPTIME-complete in the size of the domain;*
- *2EXPTIME-complete in the size of the goal.*

It is interesting to observe that if the goal is a reachability goal of the form $\Diamond G$ then the cost wrt the goal becomes polynomial (since it amounts to propositional evaluation). Moreover if for a given LTL$_f$/LDL$_f$ goals the determinization step does not cause a state explosion (which, as mentioned, is often the case) the complexity wrt the goal is EXPTIME.

4 FOND planning in fair environments

Consider *nondeterministic domains* $\mathcal{D} = (2^{\mathcal{F}}, \mathcal{A}, s_0, \delta, \alpha)$ (as before) although now assume a *fair* environment [Cimatti *et al.*, 2003; Daniele *et al.*, 1999; Pistore and Traverso, 2001; Geffner and Bonet, 2013; Sardiña and D'Ippolito, 2015]. Such a system evolves as before except the environment is assumed to satisfy a fairness assumption, which says (roughly speaking) that if an action a is applied in a state s again and again, then every possible effect will eventually happen.

Formally, a (finite or infinite) trace $\tau = s_0, a_0, s_1, a_1, \ldots$ of $\mathcal{D}$ is *fair* if for every (s, a) that occurs infinitely often in τ, if $(s, a, s') \in \delta$ then also (s, a, s') occurs infinitely often in τ. Note, in particular, that every finite trace is fair.

Given a nondeterministic domain $\mathcal{D}$ and an LTL$_f$/LDL$_f$ formula φ, a strategy f is a *fair solution in* $\mathcal{D}$ *for goal* φ if every fair f-trace of $\mathcal{D}$ starting in the initial state is finite and satisfies φ. Note that strategy f does not impose any constraints on infinite traces that are unfair.

Relationship with FOND$_{fair}$ for reachability goals. When the goal is $\Diamond G$ we get the standard form for FOND$_{fair}$, which is known to be EXPTIME-complete in the size of the set of fluents $\mathcal{F}$ [Rintanen, 2004]. There is an equivalent definition of solving FOND$_{fair}$ for reachability goals that can be expressed in the temporal logic CTL [Daniele *et al.*, 1999; Pistore and Traverso, 2001; Cimatti *et al.*, 2003]. It captures the intuition that the agent can ensure that, if the goal has not yet been reached, then should the environment ever choose to co-operate, the goal could be reached. Informally, a strategy f is a *strong-cyclic solution for* G if every state reachable using f can be extended, again using f, to a terminal state satisfying G. Equivalence between *strong-cyclic solutions* and *fair solutions* is discussed in [Sardiña and D'Ippolito, 2015].

Relationship with Probabilistic Planning. We now show how to give a probabilistic interpretation to fair solutions. This connection has been noted for reachability goals [Rintanen, 2004]. A *probabilistic domain* (including initial state) is a tuple $\mathcal{D}_p = (2^{\mathcal{F}}, \mathcal{A}, s_0, p, \alpha)$ where: $\mathcal{F}$ is a set of *fluents* (atomic propositions); $\mathcal{A}$ is a set of *actions*; $2^{\mathcal{F}}$ is the set of *domain states*; s_0 is the *initial state* (initial assignment to fluents); $\emptyset \neq \alpha(s) \subseteq \mathcal{A}$ represents *action preconditions* (we assume no dead-ends); and $p : 2^{\mathcal{F}} \times \mathcal{A} \to Dbn(2^{\mathcal{F}})$ maps pairs (s, a) with $a \in \alpha(s)$ to a probability distribution on domain states. Such a system evolves (forever) as follows: from a given state s, the agent chooses an action $a \in \alpha(s)$, after which the environment chooses a successor state s' according to the distribution $p(s, a)$.

A *strategy* for a probabilistic domain is a total function $f : (2^{\mathcal{F}})^+ \to \mathcal{A}$ such that for every $u \in (2^{\mathcal{F}})^+$ we have that $f(u) \in \alpha(last(u))$. A strategy f induces a probability distribution $Pr_{f,\mathcal{D}_p}$ over the set of all infinite traces of $\mathcal{D}_p$ [Puterman, 2005]. Thus, for a measurable set X of infinite traces, we write $Pr_{f,\mathcal{D}_p}(X)$ for the probability of the set of f-traces in $\mathcal{D}_p$ that are in X (all our sets are seen to be measurable). An infinite trace of $\mathcal{D}_p$ is *fair* if for every (s, a) that occurs infinitely often in τ, if $p(s, a)(s') > 0$ then also (s, a, s') occurs infinitely often in τ. A strategy is *finite-state* if it is induced by a finite-state Moore machine. If f is finite-state, writing R for the set of fair f-traces of $\mathcal{D}_p$, we have $Pr_{f,\mathcal{D}_p}(R) = 1$ [Kemeny and Snell, 1976].

We now describe how to translate fair domains to probabilistic domains. Given a FOND*fair* $\mathcal{D}$ define a probabilistic domain $\mathcal{D}_p$ as follows: add a fluent $done$, and an action `win!` that has no preconditions, and its effects are to make the fluent $done$ true; add $\neg done$ to the precondition of every other action; replace δ with any probability distribution $p : 2^{\mathcal{F}} \times \mathcal{A} \to Dbn(2^{\mathcal{F}})$ such that $p(s, a)(s') > 0$ iff $(s, a, s') \in \delta$. Given a finite trace τ of $\mathcal{D}$ that ends in a state s, define the infinite trace $ext(\tau) = \tau \cdot (win! \cdot s)^\omega$ of $\mathcal{D}_p$. Observe that $ext(\tau)$ is fair in $\mathcal{D}_p$. For a set X of finite traces of $\mathcal{D}$, define the set $ext(X) = \{ext(\tau) : \tau \in X\}$ of infinite traces of $\mathcal{D}_p$.

We say that an infinite trace τ of $\mathcal{D}_p$ *satisfies* an LTL$_f$/LDL$_f$ formula φ if $\tau = ext(\tau')$ for some finite trace τ' of $\mathcal{D}$, and τ' satisfies φ. Let $[\varphi]$ denote the set of infinite traces of $\mathcal{D}_p$ satisfying φ. There is a tight correspondence between strategies in $\mathcal{D}$ and in $\mathcal{D}_p$.

Theorem 2 *Let $\mathcal{D}$ be a nondeterministic domain, φ an LTL$_f$/LDL$_f$ formula, and consider finite-state strategies f in $\mathcal{D}$ and g in $\mathcal{D}_p$ that correspond. Then, f is a fair solution to $\mathcal{D}$ for goal φ iff $Pr_{g,\mathcal{D}_p}([\varphi]) = 1$.*

Proof. *Left to Right:* Suppose f is a fair solution. Let R be the set of all fair f-traces in $\mathcal{D}$ starting in the initial state. By assumption, all traces in R are finite and satisfy φ, thus $ext(R) \subseteq [\varphi]$. Let R' be the set of fair g-traces. By remark above, $Pr_{g,\mathcal{D}_p}(R') = 1$. Since $Pr_{g,\mathcal{D}_p}(\cdot)$ is monotone, it is sufficient to prove that $R' \subseteq ext(R)$. So, aiming for a contradiction, let $\tau \in R' \setminus ext(R)$. But τ is an infinite trace that doesn't mention $win!$, and thus (by definition of $g, \mathcal{D}_p$) is also an f-trace. Moreover, since τ is fair in $\mathcal{D}_p$ it is also fair in $\mathcal{D}$. This contradicts that every fair f-trace in $\mathcal{D}$ is finite.

Right to Left: Suppose $Pr_{g,\mathcal{D}_p}([\varphi]) = 1$ and τ is an f-trace of $\mathcal{D}$ starting in the initial state that does not satisfy φ. If τ is finite then $Pr_{g,\mathcal{D}_p}(\{ext(\tau)\}) > 0$ contradicting the assumption that $Pr_{g,\mathcal{D}_p}([\varphi]) = 1$. If τ is infinite then τ is also a fair g-trace in $\mathcal{D}_p$. Thus, it reaches a bottom strongly-connected component C of the finite-state Markov chain induced by $\mathcal{D}_p$ and g. In no state of C is $done$ true. Thus, every infinite trace of $\mathcal{D}_p$ that reaches C does not satisfy φ. But the probability of reaching C is non-zero, contradicting $Pr_{g,\mathcal{D}_p}([\varphi]) = 1$. ■

Along the lines of [Brafman *et al.*, 2018] we can use this theorem to reduce solving FOND*fair* with LTL$_f$/LDL$_f$ goals to solving MDPs, i.e., take the product of the DFA A_φ with $\mathcal{D}_p$ to get an equivalent MDP $A_\varphi \times \mathcal{D}_p$ with reachability goal G, and use standard MDP planners to decide "almost sure reachability", i.e., if there is a policy π ensuring the probability of reaching G is equal to 1. On the other hand, almost sure reachability can be solved by a nested fixpoint construction [de Alfaro *et al.*, 2007], which we use in our automata-theoretic technique.

Automata-theoretic solution. We give a direct automata-theoretic technique for solving FOND*fair* for LTL$_f$/LDL$_f$ (again, complexities are wrt explicit representation of $\mathcal{D}$).

```
Algorithm 2: FOND_fair for LDL_f/LTL_f goals
Given LTL_f/LDL_f domain $\mathcal{D}$ and goal $\varphi$
1: Compute DFA corresponding to $\mathcal{D}$ (poly)
2: Compute NFA for $\varphi$ (exp)
3: Determinize NFA to DFA (exp)
4: Compute product with DFA of $\mathcal{D}$ (poly)
5: Solve fair DFA game (poly)
6: Return winning strategy if one exists
```

The steps are the same as for FOND*unr*, except that the resulting game is a *fair* DFA *game* instead of a DFA game.

Fair DFA Games. A *fair* DFA *game* $\mathcal{G}$ consists of the same components as a DFA game $(2^{\mathcal{F}\cup\mathcal{A}}, S, s_{init}, \varrho, F)$. However, the notion of winning is different: a strategy f is *winning for the fair* DFA *game* $\mathcal{G}$ if every fair f-trace starting in the initial state terminates in a state of F. The idea for winning such games is that the agent should remain in a "safe area" from which it is possible to cooperatively reach the final state.

We already defined the *universal preimage*, we now define the *existential preimage* $PreE(a, Y)$ for $a \in \mathcal{A}$ and $Y \subseteq S$ as: $\{s \in S \mid \exists E \in 2^{\mathcal{F}}, \forall s', \text{ if } s' = \varrho(s, [a, E]) \text{ then } s' \in Y\}$. Let $PreAE(X, Y) = \{s \in S : \exists a \in \mathcal{A}. s \in PreA(a, X) \cap PreE(a, Y)\}$ and define two nested fixpoints, a *greatest* (for safety) and *least* (for reachability):

$$Safe = \nu X.\mu Y.F \cup PreAE(X, Y).$$

This gives rise to a nested fixpoint computation: $X_0 = S$ (all states of $\mathcal{G}$); $X_{i+1} = \mu Y.F \cup PreAE(X_i, Y)$; $Safe = \bigcap_{i\leq|S|} X_i$; where $\mu Y.F \cup PreAE(X_i, Y)$ is computed as $Y_{i,1} = F$; $Y_{i,j+1} = Y_{i,j} \cup PreAE(X_i, Y_{i,j})$; $Y_i = \bigcup_{j\leq|S|} Y_{i,j}$. Note. Computing $Safe$ is *quadratic* in the number of states of $\mathcal{G}$.

We can stratify $Safe$ according to when a state enters the least fixpoint: $Reach_1 = F$, $Reach_{j+1} = Reach_j \cup PreAE(Safe, Reach_j)$. Note that $Safe = \cup_{j\leq|S|} Reach_j$. Define $\omega : Safe \setminus F \to 2^{\mathcal{A}}$ as mapping $s \in Reach_{j+1} \setminus Reach_j$ to the set of all $a \in \mathcal{A}$ such that $\exists E.\varrho(s, [a, E]) \in Reach_j$.

Theorem 3 *For a state $s \in S$, we have $s \in Safe$ iff there is a winning strategy for the fair* DFA *game $\mathcal{G}$ starting in state s.*

Proof (sketch). Left to Right: Let $f : \mathit{Safe} \setminus F \to \mathcal{A}$ be a function, restricting $\omega(s)$, i.e., such that $f(s) \in \omega(s)$. Let $\mathcal{G}_f$ be the DFA formed from $\mathcal{G}$ by keeping transitions of the form $(s, [f(s), E], s')$, i.e., $\mathcal{G}_f$ is the game in which the player commits to playing f. We use the following properties of $s \in \mathit{Safe}$: (i) every path from s in $\mathcal{G}_f$ is in Safe until it reaches F, (ii) there is a path from s in $\mathcal{G}_f$ that reaches F.

So, let τ be a fair f-trace in $\mathcal{G}$. Then τ is a trace in $\mathcal{G}_f$ with the following property ($\star$): if s, s' are in the same strongly connected component of $\mathcal{G}_f$ and τ visits s infinitely often then τ visits s' infinitely often (indeed, induct on the length n of the shortest path from s to s': if $n = 0$ we are done; if $n > 0$ then there is an edge $(s, [f(s), E], t)$ in G_f such that the length from t to s is $< n$, but by fairness t is visited infinitely often). Thus, τ eventually reaches a bottom strongly connected component C of $\mathcal{G}_f$ (since $\mathcal{G}_f$ is finite). By (i) and ($\star$) $C \subseteq \mathit{Safe}$, and by (ii) and ($\star$) $C \cap F \neq \emptyset$. Thus, by ($\star$) the trace τ reaches F, as required.

Right to Left: We can stratify the complement of Safe according to when a state is thrown out of the greatest fixpoint. E.g., if $s \in X_0 \setminus X_1$ then there s cannot reach F, and if $s \in X_1 \setminus X_0$ then s cannot reach F without the environment moving to a state that cannot reach F, etc. In other words, from $s \notin \mathit{Safe}$, no matter what the strategy f is, after finitely many steps a trace will visit a state from which there is no path to F. Thus there are fair f-traces that never visit F. ■

Correctness and complexity analysis. As before, correctness follows from the tight correspondence between strategies in $\mathcal{D}$ and strategies in $A_{\mathcal{D}} \times A_{\varphi}$. Regarding complexity, our automata-theoretic technique gives us that solving FOND*fair* for LTL$_f$/LDL$_f$ goals is in EXPTIME wrt the domain and 2EXPTIME wrt the goal. These bounds are tight:

Theorem 4 *Solving* FOND*fair for* LTL$_f$/LDL$_f$ *goals is:*

– *EXPTIME-complete in the size of the domain;*

– 2*EXPTIME-complete in the size of the goal.*

Proof (sketch). It remains to argue the lower bounds. EXPTIME-hardness wrt the domain comes EXPTIME-hardness of FOND*fair* for standard reachability goals. Also, 2EXPTIME-hardness wrt the goal follows from 2EXPTIME-hardness wrt the goal of FOND*unr*. Indeed, let $\mathcal{D}$ be a nondeterministic domain and φ an LTL$_f$ goal. Build a planning domain $\mathcal{D}'$ which simulates $\mathcal{D}$ but has an additional counter that increases with every action until it reaches $2^{|\mathcal{F}|} + 1$, at which point no action is available. Let f be a strong solution to $\mathcal{D}$. Note that every f-trace visits each state of $\mathcal{D}$ at most once (for otherwise there would be an infinite f-trace). Thus, f is also a fair solution to $\mathcal{D}'$ for φ. Conversely, let g be a fair solution to $\mathcal{D}'$ for g. Then every fair g-trace is finite and satisfies φ. But all traces of $\mathcal{D}'$ are fair and finite. Thus g is a strong solution to $\mathcal{D}$. ■

Actually, we can strengthen this and give lower bounds on the solution memory:

Theorem 5 *Memory for strategies solving* FOND*unr and* FOND*fair for* LTL$_f$ *(and hence* LDL$_f$*) goals may be required to be doubly-exponential in the size of the goal.*

Proof (sketch). We present the proof for fair solutions (it can be adapted to strong solutions). Following [Kupferman and Vardi, 2005; Schewe, 2006], we consider the language $\mathcal{L}_n$:

$$\{\{0,1,\#\}^* \cdot \# \cdot w \cdot \# \cdot \{0,1,\#\}^* \cdot \$ \cdot w \cdot \# \cdot \#^* \mid w \in \{0,1\}^n\}$$

A word is in $\mathcal{L}_n$ iff the suffix of length n that comes after the $\$$ appears somewhere before the $\$$. The smallest DFA that accepts $\mathcal{L}_n$ has at least 2^{2^n} states [Chandra *et al.*, 1981]. This is because reaching the marker $\$$, the DFA should remember the possible set of words in $\{0,1\}^n$ that have appeared before. We can specify $\mathcal{L}_n$ with an LTL$_f$ formula $\Phi_{\mathcal{L}_n}$ of length quadratic in n. The formula $\Phi_{\mathcal{L}_n}$ makes sure that there is only one $\$$ in the word and that eventually there exists a position in which $\#$ is true and the i-th letter from this position, for $1 \leq i \leq n$, agrees with the i-th letter after the $\$$, namely:

$$[\neg\$\,\mathcal{U}(\$ \wedge \circ(0 \vee 1) \wedge \circ^2(0 \vee 1) \wedge \cdots \circ^n((0 \vee 1) \wedge \circ\Box\#)]$$
$$\wedge\Diamond[\# \wedge \textstyle\bigwedge_{1 \leq i \leq n}(\circ^i 0 \wedge \Box(\$ \supset \circ^i 0) \vee \circ^i 1 \wedge \Box(\$ \supset \circ^i 1))]$$

Define a planning domain with actions *step*, *accept*, *reject*, and fluents $\mathit{Stop}, 0, 1, \#, \$$. Initially $\neg\mathit{Stop}, \neg\mathit{Accepted}, \neg\mathit{Rejected}$ hold. The precondition of *step* is $\neg$*Stop*, and its nondeterministic effects make one of 0, 1, $\#$ $\$$ true (and the rest false) and, simultaneously, either *Stop* or $\neg$*Stop*. The precondition of *accept* (resp. *reject*) is *Stop*, and its effect is *Accepted* (resp. *Rejected*). The goal is:

$$(\Phi_{\mathcal{L}_n} \supset \Diamond\mathit{Accepted}) \wedge (\neg\Phi_{\mathcal{L}_n} \supset \Diamond\mathit{Rejected}).$$

While $\neg$*Stop* holds (as it does initially), the only possible action is *step*. By fairness, eventually *Stop* holds and the actions *accept* and *reject* can be used to set either *Accepted* or *Rejected* to be true. The goal requires that if the produced trace is in $\mathcal{L}_n$ (resp. not in $\mathcal{L}_n$) then the agent played *accept* (resp. *reject*). Intuitively, to win, the strategy needs to memorize all possible words of length 2^n, hence it cannot be smaller than 2^{2^n}. In particular, a winning strategy executed in the domain becomes a DFA for recognizing $\mathcal{L}_n$ but this cannot be smaller than 2^{2^n}. ■

5 Conclusions

We conclude by observing that DFA games with or without fairness can be seen as variants of FOND*fair* and FOND*unr* problems for reachability goals. Hence, heuristics, knowledge and solvers developed by the planning community for the latter can immediately be applied. This is a crucial point for effectiveness and scalability of the techniques.

We remark that although the 2EXPTIME bound is high, in many cases the actual complexity is much smaller, e.g., it is polynomial for $\Diamond G$, i.e., for reachability of a propositional goal G. Moreover, often the determinization step in the technique, which is in general exponential, is not needed if the automaton corresponding to the formula is already deterministic. This is, e.g., the case for the LTL$_f$ formulas used in declarative business process modeling [van der Aalst *et al.*, 2009], which are Boolean combinations of patterns that correspond to very small deterministic automata. Also, even when determinization cannot be avoided, it is often the case that it does not produce an exponential blow-up as shown in [Tabakov and Vardi, 2005]. Finally, the experiments in [Camacho *et al.*, 2017b] confirm experimentally actual feasibility in spite of 2EXPTIME-completeness.

References

[Alur *et al.*, 2002] Rajeev Alur, Thomas A. Henzinger, and Orna Kupferman. Alternating-time temporal logic. *J. of the ACM*, 49(5):672–713, 2002.

[Armoni et al., 2002] Armoni et al. The ForSpec temporal logic: A new temporal property-specification language. In *TACAS*, 2002.

[Bacchus and Kabanza, 2000] Fahiem Bacchus and Froduald Kabanza. Using temporal logics to express search control knowledge for planning. *Artif. Intell.*, 116(1-2):123–191, 2000.

[Baier and McIlraith, 2006] Jorge A. Baier and Sheila A. McIlraith. Planning with first-order temporally extended goals using heuristic search. In *Proc. of AAAI*, 2006.

[Baier *et al.*, 2008] Jorge A. Baier, Christian Fritz, Meghyn Bienvenu, and Sheila A. McIlraith. Beyond classical planning: Procedural control knowledge and preferences in state-of-the-art planners. In *AAAI*, 2008.

[Brafman *et al.*, 2018] Ronen I. Brafman, Giuseppe De Giacomo, and Fabio Patrizi. LTL$_f$/LDL$_f$ non-markovian rewards. *AAAI*, 2018.

[Camacho *et al.*, 2017a] Alberto Camacho, Oscar Chen, Scott Sanner, and Sheila A. McIlraith. Non-markovian rewards expressed in LTL: guiding search via reward shaping. In *SOC*, pages 159–160, 2017.

[Camacho *et al.*, 2017b] Alberto Camacho, Eleni Triantafillou, Christian Muise, Jorge A. Baier, and Sheila McIlraith. Non-deterministic planning with temporally extended goals: LTL over finite and infinite traces. In *AAAI*, 2017.

[Chandra *et al.*, 1981] A. K. Chandra, D. C. Kozen, and L. J. Stockmeyer. Alternation. *J. of the ACM*, 28(1), 1981.

[Cimatti *et al.*, 2003] Alessandro Cimatti, Marco Pistore, Marco Roveri, and Paolo Traverso. Weak, strong, and strong cyclic planning via symbolic model checking. *Artificial Intelligence*, 1–2(147), 2003.

[Daniele *et al.*, 1999] Marco Daniele, Paolo Traverso, and Moshe Y. Vardi. Strong cyclic planning revisited. In *ECP*, pages 35–48, 1999.

[de Alfaro *et al.*, 2007] Luca de Alfaro, Thomas A. Henzinger, and Orna Kupferman. Concurrent reachability games. *Theor. Comput. Sci.*, 386(3):188–217, 2007.

[De Giacomo and Vardi, 2013] Giuseppe De Giacomo and Moshe Y. Vardi. Linear temporal logic and linear dynamic logic on finite traces. In *IJCAI*, 2013.

[De Giacomo and Vardi, 2015] Giuseppe De Giacomo and Moshe Y. Vardi. Synthesis for LTL and LDL on finite traces. In *IJCAI*, 2015.

[Eisner and Fisman, 2006] Cindy Eisner and Dana Fisman. *A practical introduction to PSL.* Springer, 2006.

[Fogarty *et al.*, 2013] Seth Fogarty, Orna Kupferman, Moshe Y. Vardi, and Thomas Wilke. Profile trees for Büchi word automata, with application to determinization. In *GandALF*, 2013.

[Fritz and McIlraith, 2007] Christian Fritz and Sheila A. McIlraith. Monitoring plan optimality during execution. In *ICAPS*, 2007.

[Geffner and Bonet, 2013] Hector Geffner and Blai Bonet. *A Coincise Introduction to Models and Methods for Automated Planning*. Morgan & Claypool, 2013.

[Gerevini *et al.*, 2009] Alfonso Gerevini, Patrik Haslum, Derek Long, Alessandro Saetti, and Yannis Dimopoulos. Deterministic planning in the fifth international planning competition: PDDL 3 and experimental evaluation of the planners. *Artificial Intelligence*, 173(5-6):619–668, 2009.

[Ghallab *et al.*, 2004] Malik Ghallab, Dana S. Nau, and Paolo Traverso. *Automated planning – Theory and Practice*. Elsevier, 2004.

[Hanks and McDermott, 1986] Steve Hanks and Drew V. McDermott. Default reasoning, nonmonotonic logics, and the frame problem. In *AAAI*, 1986.

[Kemeny and Snell, 1976] John G. Kemeny and James Laurie Snell. *Finite markov chains, undergraduate texts in mathematics*. Springer, 1976.

[Kupferman and Vardi, 2005] Orna Kupferman and Moshe Y. Vardi. From linear time to branching time. *ACM Trans. Comput. Log.*, 6(2):273–294, 2005.

[Pistore and Traverso, 2001] Marco Pistore and Paolo Traverso. Planning as model checking for extended goals in non-deterministic domains. In *IJCAI*, 2001.

[Pnueli and Rosner, 1989] Amir Pnueli and Roni Rosner. On the synthesis of a reactive module. In *POPL*, 1989.

[Puterman, 2005] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley, 2005.

[Rabin and Scott, 1959] Michael O. Rabin and Dana Scott. Finite automata and their decision problems. *IBM J. Res. Dev.*, 3:114–125, April 1959.

[Rintanen, 2004] Jussi Rintanen. Complexity of planning with partial observability. In *ICAPS*, 2004.

[Sardiña and D'Ippolito, 2015] Sebastian Sardiña and Nicolás D'Ippolito. Towards fully observable non-deterministic planning as assumption-based automatic synthesis. In *IJCAI*, 2015.

[Schewe, 2006] Sven Schewe. Synthesis for probabilistic environments. In *ATVA*, 2006.

[Schoppers, 1987] Marcel J. Schoppers. Universal plans for reactive robots in unpredictable environments. In *IJCAI*, 1987.

[Tabakov and Vardi, 2005] Deian Tabakov and Moshe Y. Vardi. Experimental evaluation of classical automata constructions. In *LPAR*, 2005.

[van der Aalst *et al.*, 2009] W. van der Aalst, M. Pesic, and H. Schonenberg. Declarative Workflows: Balancing Between Flexibility and Support. *Computer Science - R&D*, 23(2):99–113, 2009.

[Vardi, 2011] Moshe Y. Vardi. The rise and fall of linear time logic. In *GandALF*, 2011.