Planning for LTL_f/LDL_f Goals in Non-Markovian Fully Observable Nondeterministic Domains

Ronen I. Brafman[1] and **Giuseppe De Giacomo**[2]
[1]Ben-Gurion University, Israel
[2]Università di Roma "La Sapienza", Italy
brafman@cs.bgu.ac.il, degiacomo@diag.uniroma1.it

Abstract

In this paper, we investigate non-Markovian Nondeterministic Fully Observable Planning Domains (NMFONDs), variants of Nondeterministic Fully Observable Planning Domains (FONDs) where the next state is determined by the full history leading to the current state. In particular, we introduce TFONDs which are NMFONDs where conditions on the history are succinctly and declaratively specified using the linear-time temporal logic on finite traces LTL_f and its extension LDL_f. We provide algorithms for planning in TFONDs for general LTL_f/LDL_f goals, and establish tight complexity bounds w.r.t. the domain representation and the goal, separately. We also show that TFONDs are able to capture all NMFONDs in which the dependency on the history is "finite state". Finally, we show that TFONDs also capture Partially Observable Nondeterministic Planning Domains (PONDs), but *without* referring to unobservable variables.

1 Introduction

Most formalisms for planning and reasoning about actions make the Markov assumption: the executability of an action and its effects are entirely determined by the current state [Reiter, 2001; Geffner and Bonet, 2013]. The same assumption plays a central role in Markov Decision Processes (MDPs) [Puterman, 2005]. Here, we want to consider systems with *non*-Markovian dynamics, as was done in work on reasoning about actions where executability conditions and effects of an action depend not only on what holds when the action is to occur, but also on whether certain conditions were satisfied in the past [Giunchiglia and Lifschitz, 1995; Mendez *et al.*, 1996; Gonzalez *et al.*, 2005; Gabaldon, 2011].

To get an intuition of non-Markovian dynamics, consider the following example taken from [Gabaldon, 2011]: *Imagine a robot that works in a biological research facility with different safety-level areas. The dynamics is such that a material will be considered contaminated after the robot touches it if the robot has been to a low safety area or has directly been in contact with a hazardous material, and has not been to the disinfection station since then. So the effect of touching the material depends on the history of robot activities.*

Situations in which the dynamics of the domain depend on the history are common. In simple cases, it is not difficult to avoid explicit reference to the whole history by extending the state with suitable auxiliary state variables that preserve the necessary information from the history. But in general it may not be obvious how to do so, and the resulting Markovian representation of the domain may be substantially more complex, with a larger number of state variables and corresponding description of their dynamics. In fact, as we show in this paper, this may require, in the worst case, a doubly exponential blowup in the number of states w.r.t. a non-Markovian representation expressed using temporal logic (this answers an informal conjecture in [Gabaldon, 2011]). This potential blowup makes it implausible that a human designer can come up with the suitable control variables to describe the dynamic as a Markovian system in the general case. Moreover, making intuitive sense of these auxiliary state variables may be difficult or even impossible.

In this paper we consider non-Markovian specification in the context of planning in Fully Observable Nondeterministic Domains (FOND), see e.g., [Geffner and Bonet, 2013], and investigate strong planning for temporally extended goals expressed in linear-time-logics over finite traces, namely LTL_f, and its extension LDL_f [De Giacomo and Vardi, 2013; Camacho *et al.*, 2017; De Giacomo and Rubin, 2018].

For concreteness, we introduce Temporal FONDs (or TFONDs) as a means to succinctly and declaratively define non-Markovian dynamics in planning domain, based on the use of LTL_f/LDL_f. We provide the following results:

- We devise a technique for planning in LTL_f/LDL_f TFONDs for LTL_f/LDL_f temporally extended goals. This technique is based on a reduction to a (larger) standard FOND planning problem, and hence can exploit the highly optimized FOND planners (such as NDP [Alford *et al.*, 2014], FIP [Fu *et al.*, 2016], MyND [Mattmüller *et al.*, 2010], Gamer [Kissmann and Edelkamp, 2011], PRP [Muise *et al.*, 2012], and FOND-SAT [Geffner and Geffner, 2018]).
- We caracterize the computational complexity of planning for LTL_f/LDL_f goals in TFOND as 2EXPTIME-complete both in the domain and in the goal.
- We also show that TFONDs are able to capture all Non-Markovian Nondeterministic Fully Observable Planning

Domains (NMFONDs) in which the dependency on the history is "finite state" in a precise formal sense.

Finally we look at partially observable nondeterministic planning domains (PONDs) and we show that these are equivalent in a strong sense to LDL_f TFONDs defined over the *observable* variables *only*. This result is quite interesting since it removes completely the unobservable variables, whose nature is ambiguously being postulated by the modeller, but may be intangible in reality. The ability to remove the dependence on unobservable variables is likely to help learn such models from data, which typically consists of observable entities only.

2 Preliminaries

We breifly recall the main notions regarding LTL_f/LDL_f and FOND planning. LTL_f is the *Linear-time Temporal Logic* LTL *interpreted over finite traces* [De Giacomo and Vardi, 2013], instead of infinite ones [Pnueli, 1977]. Given a set $\mathcal{P}$ of propositional symbols, LTL_f formulas φ are:

$$\varphi \quad ::= \quad \phi \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \bigcirc\varphi \mid \varphi_1 \,\mathcal{U}\, \varphi_2$$

where ϕ is a propositional formula over $\mathcal{P}$, $\bigcirc$ is the *next* operator and $\mathcal{U}$ is the *until* operator. We use the usual abbreviations such as $\varphi_1 \vee \varphi_2 \doteq \neg(\neg\varphi_1 \wedge \neg\varphi_2)$; *eventually* as $\Diamond\varphi \doteq true\,\mathcal{U}\,\varphi$; *always* as $\Box\varphi \doteq \neg \Diamond \neg\varphi$; and $\bullet\varphi \doteq \neg\bigcirc\neg\varphi$. (Note that on finite traces $\neg\bigcirc\varphi \not\equiv \bigcirc\neg\varphi$.) LTL_f is as expressive as FO (first order logic) over finite traces and star-free regular expressions (RE), thus strictly less expressive than RE, which in turn are as expressive as monadic second order logic over finite traces. RE themselves are not convenient for expressing temporal specifications, since they miss direct constructs for negation and conjunction, and adding them makes reasoning non-elementary, see e.g., [Lodaya, 2012].

For this reason, [De Giacomo and Vardi, 2013] introduced LDL_f, *linear dynamic logic on finite traces*, which merges LTL_f with RE, through the syntax of the well-known logic of programs PDL, *propositional dynamic logic* [Fischer and Ladner, 1979; Harel *et al.*, 2000; Vardi, 2011], but interpreted over finite traces. Here, following, [Brafman *et al.*, 2018], we consider a variant of LDL_f that works also on empty traces:

$$\begin{array}{lcl} \varphi & ::= & tt \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \langle\varrho\rangle\varphi \\ \varrho & ::= & \phi \mid \varphi? \mid \varrho_1 + \varrho_2 \mid \varrho_1; \varrho_2 \mid \varrho^* \end{array}$$

where tt stands for logical truth; ϕ is a propositional formula over $\mathcal{P}$ (including the propositional formula $true$, not to be confused with tt); ϱ denotes path expressions, which are RE over propositional formulas ϕ, and the test construct $\varphi?$ typical of PDL. We abbreviate $[\varrho]\varphi \doteq \neg\langle\varrho\rangle\neg\varphi$ as in PDL, $ff \doteq \neg tt$ for false, and $\phi \doteq \langle\phi\rangle tt$ to denote an occurence of propositional formula ϕ. Intuitively, $\langle\varrho\rangle\varphi$ states that, from the current step in the trace, there exists an execution satisfying the RE ϱ such that its last step satisfies φ, while $[\varrho]\varphi$ states that, from the current step, all executions satisfying the RE ϱ are such that their last step satisfies φ. Tests are used to insert into the execution path checks for satisfaction of additional LDL_f formulas. As mentioned LDL_f is as expressive as MSO over finite words. It captures LTL_f, by seeing *next* and *until* as the abbreviations $\bigcirc\varphi \doteq \langle true\rangle(\varphi \wedge \neg end)$ and $\varphi_1 \,\mathcal{U}\, \varphi_2 \doteq \langle(\varphi_1?; true)^*\rangle(\varphi_2 \wedge \neg end)$, and any RE r, with the formula $\langle r\rangle end$, where $end \doteq [true]ff$ expresses that the trace has ended. Note that in addition to end we can also denote the last element of the trace as $last \doteq \langle true\rangle end$.

An NFA is a tuple $\mathcal{A} = \langle \Sigma, Q, q_0, \delta, F\rangle$, where: (i) Σ is the input alphabet of the automaton; (ii) Q is the finite set of states; (iii) $q_0 \in Q$ is the initial state; (iv) $\delta \subseteq Q \times \Sigma \times Q$ is the transition relation; (v) $F \subseteq Q$ is the set of final states. A DFA is an NFA where δ is a function $\delta : Q \times \Sigma \to Q$. By $\mathcal{L}(\mathcal{A})$ we mean the set of all traces over Σ accepted by $\mathcal{A}$.

We can associate each LTL_f/LDL_f formula φ with an (exponentially large) NFA $\mathcal{A}_\varphi = \langle 2^{\mathcal{P}}, Q, q_0, \delta, F\rangle$ that accepts exactly the traces satisfying φ. The construction relies on the fact that (*i*) every LDL_f formula φ can be associated with a polynomial *alternating automaton on words* (AFW) accepting exactly the traces that satisfy φ [De Giacomo and Vardi, 2013], and (*ii*) every AFW can be transformed into an NFA, see, e.g., [De Giacomo and Vardi, 2013].

The NFA $\mathcal{A}_\varphi$ can be transformed into a DFA, in exponential time, and then possibly put in (the unique) minimal form, in polynomial time [Rabin and Scott, 1959]. Thus, we can transform any LTL_f/LDL_f formula into a DFA of double exponential size. While this is a worst-case complexity, in most cases the size of the DFA is actually manageable [Tabakov and Vardi, 2005]. Moreover, determinization can be performed *on-the-fly*, without the need to explicitly construct the DFA by progressing all possible states the NFA can be in after consuming the next trace symbol. We accept the trace iff, once it has been completely read, the set of possible states contains a final state. This on-the-fly construction allows for a compact (i.e., logarithmically factorized) representation of the DFA, and it is, e.g., exploited in [Camacho *et al.*, 2018a; Brafman *et al.*, 2018].

Fully Observable Nondeterministic Domains (FOND) are commonly studied in Planning see e.g., [Geffner and Bonet, 2013]. Here we consider the domains to be rooted, i.e., they include the initial state and formally define FOND as a tuple $\mathcal{D} = (P, A, 2^P, s_0, tr)$ where: 1. P is a finite set of atomic propositions also called *fluents*; 2. A is a finite set of *actions*; 3. 2^P is the set of domain states; 4. s_0 is the initial state (initial assignment to fluents); 5. $(s, a, s') \in tr$ represents *action effects (including frame assumptions)*, and implicitly also actions preconditions: action a is possible in state s if $(s, a, s') \in tr$, for any s'. 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* P, i.e., logarithmic in the number of states. Intuitively, a nondeterministic domain evolves as follows: from a given state s, the agent chooses a possible action a (i.e., such that there exists an s' such that $(s, a, s') \in tr$), after which the environment chooses a successor state s' with $(s, a, s') \in tr$. The agent can choose its action based on the history of states so far (i.e., the agent has full observation).

Given a domain $\mathcal{D}$ and a LTL_f/LDL_f *goal* formula φ over fluents P, 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. A solution this problem is given in [De Giacomo and Rubin, 2018] (see also [Camacho *et al.*, 2018a; 2018b]):

Solving FOND for LTL_f/LDL_f goals
Given FOND $\mathcal{D}$ and goal φ_G
1: Compute DFA $\mathcal{A}_{\varphi_G}$ for φ_G *(2exp in φ_G)*
2: Compute product FOND $\mathcal{D}_p$ of $\mathcal{D}$ and $\mathcal{A}_{\varphi_G}$ *(poly)*
3: Solve FOND $\mathcal{D}_p$ for goal $g = \{(s,q) \mid q \in F_G\}$
(poly in the number of states of $\mathcal{D}_p$)

Here the *product FOND* $\mathcal{D}_p = (P_p, A, S_p, s_{p,0}, tr_p)$ of a FOND $\mathcal{D} = (P, A, 2^P, s_0, tr)$ and a goal DFA $\mathcal{A}_{\varphi_G} = (2^P, Q_G, q_{G,0}, \delta_G, F_G)$ is defined as follows: (*i*) $P_p = P \cup P_{Q_G}$, where P_{Q_G} is a set of additional fluents for the logarithmic representation of Q_G, (*ii*) $S_p = 2^P \times Q_G$, (*iii*) $((s,q), a, (s',q')) \in tr_p$ iff $(s, a, s') \in tr$ and $\delta_G(q, s') = q'$ (where, we project s' on the propositions in P only). We have following complexity characterization (note that the complexity in the domain is the same as in classical FOND [Rintanen, 2004]):

Theorem 1. *[De Giacomo and Rubin, 2018] Planning for LTL_f/LDL_f goals in FONDs is: EXPTIME-complete in the domain (compactly represented) and 2EXPTIME-complete in the goal.*

3 NMFONDs

We now introduce the basic semantic model of a Non-Markovian Nondeterministic Fully Observable Planning Domain (NMFOND). An NMFOND is very much like a standard FOND, except that the effect of an action depends on the entire history of states (trace) of the system. A *Non-Markovian Nondeterministic Fully Observable Planning Domain (NMFOND)* is a tuple $\mathcal{D} = (P, A, S, s_0, tr)$, where 1. P is finite set of primitive propositions (the *fluents*); 2. A is a finite set of *actions*; 3. $S = 2^P$ is the set of possible truth assignments to P, which we still call *domain states*; 4. $s_0 \in S$ denotes the initial state (with an initial dummy action a_0); 5. $tr : S^+ \times A \times S$ is a transition relation, such that $((s_0, \ldots, s_k), a, s') \in tr$ if given the sequence of states $s_0, ..., s_k$ starting from the initial state s_0 and an action a, the state s' is a possible outcome resulting from executing the action a after traversing $s_0, ..., s_k$. (We use the standard notation S^+ for denoting $S \cup S{\times}S \cup S{\times}S{\times}S \cdots$.) Note that if $tr : S^+ \times A \times S$ has in fact the form $tr : S \times A \times S$, i.e., instead of the sequence $s_0, \ldots, s_k$ we consider only one state s, then the NMFOND is actually a standard FOND.

A *goal* for an NMFOND is a subset G of S^+, denoting the desired traces. A *plan* is a partial function $p : S^+ \to \mathcal{A}$ such that for every $w \in S^+$, if $p(w)$ is defined then $\exists s'.(w, p(w), s') \in tr$ (the action chosen by $p(w)$ is executable). If $p(w)$ is undefined, we write $p(w) = \bot$. A trace τ is *generated by* p, or simply a *p-trace*, if (i) for any prefix $s_0, \ldots, s_k, s_{k+1}$ of τ we have that $p(s_0, \ldots, s_k) = a$ for some action a and $((s_0, \ldots, s_k), a, s_{k+1}) \in tr$; and (ii) if τ is finite, say $\tau = s_0, \ldots, s_n$, then $p(s_0, \ldots, s_n) = \bot$.

Given an NMFOND $\mathcal{D}$ and a *goal* G, a plan p *realizes* G *in* $\mathcal{D}$ if every p-trace of $\mathcal{D}$ satisfies G.

4 TFONDs

A declarative way to specify NMFONDs is by using a logical language to describe properties of finite traces, and a natural choice for such a logical formalism is LTL_f and its extension LDL_f. Hence, we introduce Temporal FONDs (TFONDs). Formally a TFOND $\mathcal{D}_L$ is a tuple $\mathcal{D}_L = (P, A, S, s_0, tr_L)$, where P, A, S and s_0 are as NMFONDs, while the transition function is defined implicitly by tr_L as follows: tr_L is given in terms of a set T of triples of the form:

$$(\varphi, a, \phi)$$

where φ is a temporal LTL_f/LDL_f formula over P, $a \in A$, and ϕ is a propositional formula over P. Each of the triples (φ, a, ϕ) states that if the trace $s_0, \ldots, s_k$ from the initial state s_0 to the current state s_k satisfies φ then performing action a may bring about any state s' where ϕ is true. We call LTL_f (resp. LDL_f) TFONDs those TFONDs that use LTL_f (resp. LDL_f) for φ. Given a trace $s_0, \ldots, s_k$ and an action a, the possible next states are those satisfying the propositional formula:

$$\bigwedge_{(\varphi,a,\phi) \in T, s_0, \ldots, s_k \vDash \varphi} \phi$$

In other words: $((s_0, \ldots, s_k), a, s') \in tr$ iff $s' \vDash \bigwedge_{(\varphi,a,\phi) \in T, s_0, \ldots, s_k \vDash \varphi} \phi$. Note that if $\bigwedge_{(\varphi,a,\phi) \in T, s_0, \ldots, s_k \vDash \varphi} \phi_i \equiv \mathit{false}$ then there is no $s' \vDash \bigwedge_{(\varphi,a,\phi) \in T, s_0, \ldots, s_k \vDash \varphi} \phi_i$, i.e., there are no transitions at all (we can say that the precondition of the actions are violated). Otherwise there is one transition for each s' satisfying the condition. Note also that if for no φ we have $s_0, ..., s_k \vDash \varphi$ then there are no restrictions on s', i.e., the transition nondeterministically goes to every possible state.

Observe that if one restricts φ to be propositional then triples (φ, a, ϕ) get a quite standard (Markovian) meaning, similar, say, to PDDL conditional effects: if the current state s satisfies condition φ then by doing action a one brings about effect ϕ. Note that, differently from PDDL, we need to specify persistence explicitly using further triples. This is easy to do when capturing PDDL, but having such specification derived automatically in the general case, e.g., by exploiting the semantic principle of "explanation closure" used for deriving Successor State Axioms in [Reiter, 2001], requires further studies. PDDL also allows for "axioms" that define derived predicates and can substantially simplify the representation of planning problems [Thiébaux *et al.*, 2005]. These can be introduced in TFONDs as well, and are related to state constraints [Reiter, 2001]. We leave both issues to future work. Finally, observe that, if in a TFOND we do not constrain the effects of an action a through some triple then the next state can be arbitrary. But notice that if we want to constrain the effects to be always within, say, $\phi_1 \ldots \phi_n$, similarly to a "one-of" clause in PDDL, we can simply add the triple $(true, a, \phi_1 \vee \ldots \vee \phi_n)$.

Consider the example in the introduction [Gabaldon, 2011], and suppose we have the following fluents among other: Material-is-Contaminated (mc_i, for each material i); Robot-in-Low-Safety-Area ($rlsa$); Robot-in-Contact-with-Hazardous-Material ($rchm$); Robot-in-Disinfection-Station (rds), and action robot-touched-material i ($touch_i$, for each material i) Then the (non-Markovian) effect "*a material i will be considered contaminated after the robot touches it if the robot has been to a low safety area or has directly been in contact with a hazardous material, and has not been to the disinfection station since then*" can be specified in LDL_f as:

$$(\langle true^* ; (rlsa \vee rchm) ; (\neg rds)^* \rangle end,\ touch_i,\ mc_i)$$

which says that, if trace $s_0, ..., s_k$ from the initial instant 0 to the current instant k satisfies the formula $\langle true^*; (rlsa \vee rchm); (\neg rds)^* \rangle end$, i.e., $s_0, ..., s_k$ belongs to the regular language $true^*; (rlsa \vee rchm); (\neg rds)^*$, then by doing action $touch_i$ we get to a state s_{k+1} where mc_i holds i.e., where the material i is contaminated.

5 Planning in TFONDs for LTL_f/LDL_f Goals

Next we look into solving planning in TFONDs for LTL_f/LDL_f goals. We start by observing that planning in TFONDs even for reachability goals is at least as difficult as LTL_f/LDL_f synthesis, which is 2EXPTIME-complete [De Giacomo and Vardi, 2015]. Consider a very simple TFOND $\mathcal{D}_L = (P, A, 2^P, s_0, tr_L)$ with a single triple $(\varphi, stop, Done)$, where $stop$ is an action in A and $Done$ a fluent in P, and consider as the goal reaching a state where $Done$ is true. Then there exists a (strong) plan for reaching $Done$ if and only if the agent controlling actions A has a winning strategy against the environment controlling P. So in general, planning in TFONDs can be difficult. But how can we solve it at all?

We now show that TFONDs can be translated into FONDs with additional fluents. To do so we first introduce what we call *finite NMFONDs*, and show that TFONDs can be translated into them. A finite NMFOND $\mathcal{D} = (P, A, S, s_0, tr)$ is an NMFOND whose transition relation tr is generated by a finite state transition transducer $Tr = (Q, q_0, \delta, \varrho)$: (*i*) Q is the *finite* set of the transducer states; (*ii*) q_0 is the initial state of the transducer; (*iii*) $\delta : Q \times S \to Q$ is the transducer (deterministic) transition function; (*iv*) $\varrho : Q \times A \times S$ is the output function of the transducer. As usual, we extend the transition function $\delta : Q \times S \to Q$ to sequences, i.e., $\delta : Q \times S^+ \to Q$, in the standard way [Hopcroft and Ullman, 1979].
Tr defines the NMFOND transition relation $tr : S^+ \times A \times S$ as:

$$((s_1, \ldots, s_k), a, s') \in tr \text{ iff}$$
$$\delta(q_0, (s_1, \ldots, s_k)) = q_k \text{ and } (q_k, a, s') \in \varrho$$

Thus, the information about the current trace $s_1, \ldots, s_k$ relevant to the future behavior of the NMFOND is summarized by $\delta(q_0, (s_1, \ldots, s_k))$, and the transition relation tr is represented by making use of δ together with ϱ.

Now we show that given a TFOND $\mathcal{D}_L$, we can construct an equivalent finite NMFOND $\mathcal{D}$. Let us consider the set T of formulas of the form (φ, a, ϕ) defining tr_L. For convenience let's enumerate the triples in T as $(\varphi_1, a_1, \phi_1), \ldots, (\varphi_n a_n, \phi_n)$. For each formula φ_i, we build the corresponding DFA $\mathcal{A}_i = (2^P, Q_i, q_{i,0}, \delta_i, F_i)$, i.e., an automaton that accepts exactly those traces satisfying this formula. We define a corresponding finite NMFOND $\mathcal{D} = (P, A, S, s_0, tr)$ with tr generated by the transducer $Tr = (Q, q_0, \delta, \varrho)$ with: 1. $Q = Q_0 \times \cdots \times Q_n$; 2. $q_0 = q_{1,0}, \ldots, q_{n,0}$; 3. $\delta : Q \times S \to Q$ where $\delta((q_1, \ldots, q_n), s) = (q'_1, \ldots, q'_n))$ iff $q'_i = \delta_i(q_i, s)$; 4. $\varrho : Q \times A \times S$ where $((q'_1, \ldots, q'_n), a, s') \in \varrho$ iff $s' \vDash \bigwedge_{i:(\varphi_i, a, \phi_i), q'_i \in F_i} \phi_i$. The construction is based on computing the automata for the LTL_f/LDL_f formulas in the triples, and then run all of them in parallel. The resulting NMFOND $\mathcal{D}$ is equivalent to the TFOND $\mathcal{D}_L$ in the sense that they produce sets of traces that are isomorphic (i.e., in bijection).

Next we show that given a finite NMFOND $\mathcal{D} = (P, A, S, s_0, tr)$ with tr generated by any finite transducer $Tr = (Q, q_0, \delta, \varrho)$, we can construct an equivalent FOND $\mathcal{D}_f = (P_f, A, S_f, s_{f,0}, tr_f)$ as follows: 1. $P_f = P \cup P_{aut}$ where P is the finite set of primitive propositions and P_{aut} are propositions for the separate binary encoding of states Q of the transducer Tr; 2. A is the finite set of actions of $\mathcal{D}$; 3. S_f is the set of possible truth assignments to P_f, which we call *(domain) states*; 4. $s_{f,0} = s_0 \cup q_0$, where s_0 is a truth assignment over P denoting the initial state, and q_0 is the (encoding of) the initial state of Tr; 5. $tr : S_f \times A \times S_f$, such that:

$$((s, q), a, (s', q')) \in tr_f \text{ iff } q' = \delta(q, s) \wedge (q', a, s') \in \varrho.$$

Essentially, since the aspects of the history that are relevant to future transitions in an NMFOND can be summarized by some state $q \in Q$, if we maintain both the current element of Q and of S, we have all the information that we need.

Again, the finite NMFOND $\mathcal{D}$ and the resulting finite FOND $\mathcal{D}_f$ are equivalent in the sense that they produce sets of traces that are isomorphic (i.e., in bijection). Hence, by combining above constructions we get a way of solving LTL_f/LDL_f TFOND planning.

Solving TFOND planning for LTL_f/LDL_f goals
Given a TFOND $\mathcal{D}_L$ and a goal φ_G
1: Compute the finite NMFOND $\mathcal{D}$ *(2exp in $\mathcal{D}_L$)*
2: Compute FOND $\mathcal{D}_f$ equivalent to $\mathcal{D}$ *(poly)*
3: Compute DFA $\mathcal{A}_{\varphi_G}$ for φ_G *(2exp in φ_G)*
4: Compute product FOND $\mathcal{D}_p$ of $\mathcal{D}_f$ and $\mathcal{A}_{\varphi_G}$ *(poly)*
5: Solve FOND $\mathcal{D}_p$ for goal $g = \{(s, q) \mid q \in F_G\}$
(poly in the number of states of $\mathcal{D}_p$)

Now we are ready to give the complexity characterization.

Theorem 2. *Planning for finite trace temporal goals in LTL_f/LDL_f TFONDs is 2EXPTIME-complete in the domain and 2EXPTIME-complete in the LTL_f/LDL_f goals.*

Proof (sketch). Membership comes from the construction above. Note that we easily devise a logarithmic representation in terms of fluents for $\mathcal{D}_p$, however, solving FOND with a compact representation requires exponential time. The 2EXPTIME-hardness in the domain comes for the reduction of LTL_f synthesis to TFONDs discussed above, which is for a fixed goal. The 2EXPTIME-hardness in the goal comes from planning for LTL_f goals in FONDs [De Giacomo and Rubin, 2018], which are a special case of TFONDs. □

The increase of complexity in the domain with respect to standard FOND is due to the succinctness of the representation that LTL_f/LDL_f formulas provide: the corresponding FOND requires exponentially many additional auxiliary fluents to logarithmically represent the DFAs for the formulas in the triples. This blowup is inevitable, and it makes it virtually impossible even for a human to come up directly with a FOND that captures dependencies on history that are conveniently expressed in LTL_f/LDL_f. This answers positively an informal conjecture in [Gabaldon, 2011].

Observe that, in practice, the 2EXPTIME worst-case complexity often does not manifest itself, since DFAs for LTL_f/LDL_f formulas are often small. Moreover, while, in general, deteminization of NFA can produce an exponential blowup, often the size of the DFA obtained (possibly after minimization) from a NFA, is comparable to that of the NFA itself [Tabakov and Vardi, 2005].

The reduction from TFOND to FOND can be seen as a way to synthesize automatically and conveniently "control fluents". To see this, consider the following example: an action "rest" has the effect of "putting the robot on maintenance" only when performed in an history satisfying $\langle(s;(a;b^*;c)^*;e)^*\rangle end$, i.e., a finite repetition of starting condition, followed by finite repetitions of three phases a ("preparing an item"), b ("cleaning the item"), c ("finalizing the item"), in which b can be repeated an arbitrary number of times (multiple cleaning phases may be needed), then followed by an end condition. The minimal DFA for this formula has 7 states (see, e.g., LDL_f translator at https://flloat.herokuapp.com). Hence, if we factor these states into a binary representation we get 3 new control fluents. Note that it is not easy to come with these 3 fluents in the first place since it is not obvious how to assign intuitive meaning to them (after all they simply capture the dynamics of the DFA). As the number of these formulas grows, it becomes more and more unmanageable to come up with such control fluents directly. For example, if we replace the previous formula with $\langle(s;(a;b^*;c)^*;e)^*\rangle end \wedge [true^*;(a;c;a;c)]ff$, intuitively, disallowing two consecutive repetition of phase a and c in which cleaning does not occur, the number of states in the minimal DFA grows to 32 and hence we need 5 control fluents whose dynamics is rather obscure. Note also that if we consider the two formulas $\langle(s;(a;b^*;c)^*;e)^*\rangle end$ and $[true^*;(a;c;a;cx)]ff$ (whose minimal DFA has 6 states) separately, then we get 6 control fluents in total, which is more than what is actually needed.

6 TFONDs and finite NMFONDs

We can consider finite NMFONDs as the class of NMFONDs in which the dependency of transitions on the history is finite-state (i.e., can be expressed by using a finite state function). Above, we have seen that TFONDs can be translated into finite NMFONDs. We now show every finite NMFOND can be translated into LDL_f TFONDs. Hence, LDL_f TFONDs can be considered a sort of declarative counterpart of finite NMFONDs. Note that it is not true for LTL_f TFONDs, since LTL_f does not have the full power of RE needed for the translation.

Let $\mathcal{D} = (P, A, S, s_0, tr)$ be a *finite NMFOND* with tr generated by a finite state transition transducer $Tr = (Q, q_0, \delta, \varrho)$. We define the corresponding LDL_f TFOND $\mathcal{D}_L = (P, A, S, s_0, tr_L)$, where the transition relation tr_L is generated by a set T of triples (φ, a, ϕ) defined as follows: For every transducer state q we consider the DFA $\mathcal{A}^q_{Tr}$ defined as $\mathcal{A}^q_{Tr} = (Q, q_0, \delta, \{q\})$. That is, we consider the DFA obtained from the transducer by removing the output function ϱ and making final the state q. Notice that we have one $\mathcal{A}^q_{Tr}$ for each q but all of them are identical except for the accepting state. From basic automata theory, we know that there is a regular expression r_q that accepts the paths, i.e., the traces in S^+ leading from the initial state q_0 to q. Hence we can express reaching q by the LDL_f formula $\varphi_q = \langle r_q\rangle end$.

Using these formulas, for every transducer state q and every action a, we can define the triple (φ_q, a, ϕ) where $\phi = \bigvee_{(q,a,s')\in\varrho} s'$. As observed above, all DFA $\mathcal{A}^q_{Tr}$ (for $q \in Q$) are identical except for the accepting state, hence each formula φ_q can be represented by the exact same DFA, except for the final states, which depend on the specific q. This means that when we transform such an LDL_f TFOND $\mathcal{D}_L = (P, A, S, s_0, tr_L)$ into the corresponding finite TFOND using the construction above, we get exactly $\mathcal{D} = (P, A, S, s_0, tr)$.

Theorem 3. *Every finite NMFOND can be expressed as a* LDL_f *TFOND, and vice-versa.*

7 PONDs and TFONDs

In this section we consider Partially Observable Nondeterministic Planning Domains (POND). Our main result here is to show the equivalence between POND and (fully observable) TFONDs. While it is not surprising that a POND can be translated into a FOND – this is the well known belief state construction [Bonet and Geffner, 2000; Goldman and Boddy, 1996; Reif, 1984] – the significant aspect of this representability result is that the TFOND makes no mention of the unobservable fluents, not even in the form of belief states. This is particularly significant in the case of learning, where we may have no a-priori knowledge of the nature of the hidden variables. Indeed, if we think of an agent situated in a domain, this agent has access only to its observations, and knowledge of hidden variables presupposes some prior knowledge pertaining to the understanding of the underlying structure of the domain.

We formally define POND as follows: A *Partially Observable Nondeteministic Planning Domain (POND)* is a tuple $PO = \langle O, U, A, S, S_0, tr\rangle$, where 1. $P = O \cup U$ is a finite set of atomic propositions or fluents, partitioned ($O \cap U = \varnothing$) into *observable* ones O and *unobservable* ones U; 2. A is a finite set of actions; 3. $S = 2^O \times 2^U$ is the set of states of PO which are truth assignments to P partitioned into the one for O and the one for U. We denote states by $(o, u) \in S$ 4. $S_0 = \{(o_0, u_1), \ldots, (o_0, u_n)\}$ is the set of possible initial states in S, all sharing the same observation o_0; 5. $tr : S\times A\times S$ is the transition relation. POND is a Markovian model where both observable and unobservable fluents are nondeterministic. The dynamics of the state is Markovian and depends on the current value of all the fluents – both observable and unobservable, and the current action. Initially, the acting agent knows the value of all the observable fluents, and following each action, it gets to observe their new value. Note that the fact that P is divided into fully observable and unobservable fluents does not lead to loss of generality, as it easily captures a model with separate transition and observation functions, often adopted in POMDPs [Kaelbling *et al.*, 1998].

A *goal* for a POND PO is a subset G of $(2^O)^+$, denoting the desired sequences of observations. A *plan* is a partial function $p : (2^O)^+ \to A$ such that for every $w \in S^+$, if $p(w_{|O})$ is defined (where $w_{|O}$ is the projection of w onto O) then $\exists s'.(w, p(w_{|O}), s') \in tr$ (the action chosen by $p(w_{|O})$ is executable). If $p(w_{|O})$ is undefined, we write $p(w_{|O}) = \perp$. A trace τ is *generated by* p, or simply a *p-trace*, if (i) if $s_0, \ldots s_k$ is a prefix of τ then $s_0, \ldots s_k, p((s_0, \ldots, s_k)_{|O}), s_{k+1}$ is also a prefix of τ, and (ii) if τ is finite, say $\tau = s_0, \ldots, s_n$, then $p((s_0, \ldots, s_n)_{|O}) = \perp$. Given a POND PO and a *goal* G, a plan p *realizes* G *in* PO if every p-trace of PO satisfies G. Notice that this definition is essentially the same as for NMFOND, with the only exception that the plan p takes as

input only the sequence of observations, instead of the states themselves. As a result, the choice of action at each point of time depends only on what was observed so far.

We can reduce POND to FOND by the so called *belief state construction* [Goldman and Boddy, 1996; Reif, 1984]. Given a POND $PO = \langle O, U, A, S, S_0, tr\rangle$ we can construct a FOND $\mathcal{D}_{po} = \langle O \times 2^U, A, S_{po}, b_0, tr_{po}\rangle$ defined as follows: 1. $O \times 2^U$ are the available propositions (in fact, the propositions 2^U are expressed in terms of knowledge, typically); 2. A is the original set of actions; 3. $S_{po} = 2^O \times 2^{2^U}$ is the set of states, of the form $b = (o, \mathbf{u})$, where o is a truth assignments to the observable fluents, and $\mathbf{u}$ is a set of possible truth assignments for the unobservable fluents. Each state b is a belief-state; 4. $b_0 = (o_0, \mathbf{u}_0)$ where $\mathbf{u}_0 = \{u \mid (o_0, u) \in S_0\}$ is the initial *belief state* corresponding to the set of (o_0, u) considered initially possible, which forms the initial state of PO; 5. $S_{po} \times A \times S_{po}$ is the transition function defined as follows: $((o, \mathbf{u}), a, (o', \mathbf{u}')) \in tr_{po}$ iff (*i*) for all $u \in \mathbf{u}$ there exists a u' such that $((o, u), a, (o', u')) \in tr$; and (*ii*) $\mathbf{u}' = \{u' \mid u \in \mathbf{u}$ and $((o, u), a, (o', u')) \in tr\}$; notice that given o, $\mathbf{u}$, a and o' there exists a single $\mathbf{u}'$ such that $((o, \mathbf{u}), a, (o', \mathbf{u}')) \in tr_{po}$.

Theorem 4 ([Goldman and Boddy, 1996; Reif, 1984])**.** *A plan realizes G in PO iff it realizes G in $\mathcal{D}_{po}$.*

We formalize the correspondence between POND and TFONDs whose fluents are restricted to the observable ones only. For convenience, we work with finite NMFONDs instead of TFONDs. We start by formalizing when a POND and a finite NMFOND produce the same observable behavior. A POND $PO = \langle O, U, A, S, S_0, tr\rangle$ with observable fluents O and an NMFOND $\mathcal{D} = \langle O, A, 2^O, o_0, tr_o\rangle$ with fluents O are *O-indistinguishable* iff (*i*) for every sequence of observations and actions: $o_0, a_1, o_1, \ldots, a_k, o_k$ in $\mathcal{D}$ with $((o_0, o_1, \ldots, o_{i-1}), a_i, o_i) \in tr_o$ for $i = 1, \ldots, k$, there exists a sequence of states and actions $s_0, a_1, \ldots, a_k, s_k$ in PO such that $s_0 \in S_0$, and $(s_{i-1}, a_i, s_i) \in tr$ for $i = 1, \ldots, k$, and $s_i|_O = o_i$ for $i = 0, \ldots, k$; (*ii*) for every $s_0, a_1, s_1, \ldots, a_k, s_k$ in PO with $s_0 \in S_0$ and $(s_{i-1}, a_i, s_i) \in tr$ for $i = 0, \ldots, k$, let $s_i|_O = o_i$, then $o_0, a_1, o_1, \ldots, a_k, o_k$ is such that o_0 is the initial state of $\mathcal{D}$ and $((o_0, o_1, \ldots, o_{i-1}), a_i, o_i) \in tr_o$ for $i = 1, \ldots, k$. Next we show that for every POND, there exists a *finite* NMFOND over its observables that is O-indistinguishable.

Theorem 5. *For every POND $PO = \langle O, U, A, S, S_0, tr\rangle$ there exists a finite NMFOND $\mathcal{D}_t = \langle O, A, 2^O, o_0, tr_t\rangle$ such that PO and $\mathcal{D}_t$ are O-indistinguishable.*

Proof. Given PO we start with the belief set construction, defining a standard FOND $\mathcal{D}_{po} = \langle O \times 2^U, A, S_{po}, b_0, tr_{po}\rangle$. From $\mathcal{D}_{po}$ we define the finite NMFOND $\mathcal{D}_t = \langle O, A, 2^O, o_0, tr_t\rangle$, where O are the observable fluents and A the original set of actions; $S_{po} = 2^O$ is the set of states and o_0 the initial observation; tr_t is generated by a finite state transition transducer $Tr = \langle Q, q_0, \delta, \varrho\rangle$ where: 1. $Q = \{q_0\} \cup S_{po}$ is the set of the transducer states; 2. q_0 is the initial state of the transducer; 3. $\delta : Q \times S \to Q$ is the deterministic transition function of the transducer, defined as follows: (*i*) $\delta(q_o, o_0) = (o_0, \mathbf{u}_0)$; (*ii*) $\delta((o, \mathbf{u}), o') = (o', \mathbf{u}')$ where $((o, \mathbf{u}), a, (o', \mathbf{u}')) \in tr_{po}$ with $o' \vDash a$; 4. $\varrho : Q \times A \times S$ is the output function of the transducer, defined as follows: (*i*) $(q_o, a_0, o_0) \in \varrho$ where a_0 is the (dummy) action in o_0; (*ii*) $((o, \mathbf{u}), a, o') \in \varrho$ where again $((o, \mathbf{u}), a, (o', \mathbf{u}')) \in tr_{po}$ with $o' \vDash a$. It is easy to see that $\mathcal{D}_t = \langle O, A, 2^O, o_0, tr_t\rangle$ and $\mathcal{D}_{po} = \langle O \times 2^U, A, S_{po}, b_0, tr_{po}\rangle$ are indeed O-indistinguishable. □

Also the reverse result holds.

Theorem 6. *For every finite NMFOND $\mathcal{D}$ with fluents O there exists a POND PO with observables O such that $\mathcal{D}$ and PO are O-indistinguishable.*

Proof. Take the finite NMFOND, and using the construction in the previous section, transform it into a FOND. Let O be the set of fluents that define the state of the finite NMFOND. Let U be fluents that correspond to (a boolean encoding of) the state of the finite transducer. We can easily re-express transitions of the FOND in terms of such fluents. Considering U as unobservable fluents we get the desired POND. □

Considering the relationship between finite NMFOND and TFONDs, we finally get:

Theorem 7. *For every POND PO there exists a LDL_f TFOND $\mathcal{D}_L$ such that PO and $\mathcal{D}_L$ are O-indistinguishable. Moreover, for every LDL_f/LTL_f TFOND $\mathcal{D}_L$ there exists a POND PO such that $\mathcal{D}_L$ and PO are O-indistinguishable.*

Hence, LDL_f TFONDs (but not LTL_f TFOND, in general) provide an alternative to POND with one major benefit – in TFONDs there is no mention of, nor no need to know, nor hypothesize, a set U of unobservables.

Finally we observe that the construction that reduces POND to a finite NMFOND generates a transducer Tr_t, which can be minimized by reducing its number of states, without changing the traces generated by the NMFOND, and hence the observable traces of the original POND. If we transform back the resulting finite NMFOND into a POND with new unobservable fluents which are a factorization of the minimized states of Tr_t, we get a POND that is equivalent to the original one but with a minimal number of unobservable fluents. The practical ramification of such an observation deserves further study.

8 Conclusion

In many natural cases, the dynamics of the domain of interest is non-Markovian. In this paper we introduced and studied NMFONDs and, in particular, TFONDs, which provide a succinct and elegant formalism for specifying non-Markovian dynamics and rewards. We showed that this model can be solved using one's favorite FOND algorithm after an initial automated preprocessing phase. With these tools, we believe handling non-Markovian models becomes quite realistic and undaunting.

Acknowledgements

This work is partially supported by the Israel Ministry of Science and Technology grant 877017, the Lynn and William Frankel Center for Computer Science, and Sapienza project "Immersive Cognitive Environments".

References

[Alford *et al.*, 2014] Ronald Alford, Ugur Kuter, Dana S. Nau, and Robert P. Goldman. Plan aggregation for strong cyclic planning in nondeterministic domains. *Artif. Intell.*, 216:206–232, 2014.

[Bonet and Geffner, 2000] Blai Bonet and Hector Geffner. Planning with incomplete information as heuristic search in belief space. In *AIPS*, pages 52–61. AAAI, 2000.

[Brafman *et al.*, 2018] Ronen Brafman, Giuseppe De Giacomo, and Fabio Patrizi. LTLf/LDLf non-Markovian rewards. In *AAAI*, 2018.

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

[Camacho *et al.*, 2018a] Alberto Camacho, Jorge A. Baier, Christian J. Muise, and Sheila A. McIlraith. Finite LTL synthesis as planning. In *ICAPS*, 2018.

[Camacho *et al.*, 2018b] Alberto Camacho, Christian J. Muise, Jorge A. Baier, and Sheila A. McIlraith. Synkit: LTL synthesis as a service. In *IJCAI*, pages 5817–5819, 2018.

[De Giacomo and Rubin, 2018] Giuseppe De Giacomo and Sasha Rubin. Automata-theoretic foundations of FOND planning for LTLf and LDL_f goals. In *IJCAI*, 2018.

[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.

[Fischer and Ladner, 1979] Michael J. Fischer and Richard E. Ladner. Propositional dynamic logic of regular programs. *J. Com. Systems and Science*, 18, 1979.

[Fu *et al.*, 2016] Jicheng Fu, Andres Calderon Jaramillo, Vincent Ng, Farokh B. Bastani, and I-Ling Yen. Fast strong planning for fully observable nondeterministic planning problems. *Ann. Math. Artif. Intell.*, 78(2):131–155, 2016.

[Gabaldon, 2011] Alfredo Gabaldon. Non-Markovian control in the situation calculus. *Artif. Intell.*, 175(1):25–48, 2011.

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

[Geffner and Geffner, 2018] Tomas Geffner and Hector Geffner. Compact policies for fully observable nondeterministic planning as SAT. In *ICAPS*, 2018.

[Giunchiglia and Lifschitz, 1995] Enrico Giunchiglia and Vladimir Lifschitz. Dependent fluents. In *IJCAI*, 1995.

[Goldman and Boddy, 1996] Robert P. Goldman and Mark S. Boddy. Expressive planning and explicit knowledge. In *AIPS*, 1996.

[Gonzalez *et al.*, 2005] Graciela Gonzalez, Chitta Baral, and Michael Gelfond. Alan: An action language for modelling non-markovian domains. *Studia Logica*, 79(1):115–134, 2005.

[Harel *et al.*, 2000] David Harel, Dexter Kozen, and Jerzy Tiuryn. *Dynamic Logic*. MIT Press, 2000.

[Hopcroft and Ullman, 1979] J. Hopcroft and J. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 1979.

[Kaelbling *et al.*, 1998] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artif. Intell.*, 101(1-2):99–134, 1998.

[Kissmann and Edelkamp, 2011] Peter Kissmann and Stefan Edelkamp. Gamer, a general game playing agent. *KI*, 25(1):49–52, 2011.

[Lodaya, 2012] Kamal Lodaya. A language-theoretic view of verification. In *Modern Applications of Automata Theory*, pages 149–170. World Scientific, 2012.

[Mattmüller *et al.*, 2010] Robert Mattmüller, Manuela Ortlieb, Malte Helmert, and Pascal Bercher. Pattern database heuristics for fully observable nondeterministic planning. In *ICAPS*, 2010.

[Mendez *et al.*, 1996] Gisela Mendez, Jorge Lobo, Jimena Llopis, and Chitta Baral. Temporal logic and reasoning about actions. In *3rd Symp. on Logical Formalizations of Commonsense Reasoning*, 1996.

[Muise *et al.*, 2012] Christian J. Muise, Sheila A. McIlraith, and J. Christopher Beck. Improved non-deterministic planning by exploiting state relevance. In *ICAPS*, 2012.

[Pnueli, 1977] Amir Pnueli. The temporal logic of programs. In *FOCS*, 1977.

[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.

[Reif, 1984] John H. Reif. The complexity of two-player games of incomplete information. *Journal of Computer and System Sciences*, 29(2):274 – 301, 1984.

[Reiter, 2001] Raymond Reiter. *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*. The MIT Press, 2001.

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

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

[Thiébaux *et al.*, 2005] Sylvie Thiébaux, Jörg Hoffmann, and Bernhard Nebel. In defense of PDDL axioms. *Artif. Intell.*, 168(1-2):38–69, 2005.

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