Planning for LTL_f/LDL_f Goals in Practice

Francesco Fuggitti
fuggitti@diag.uniroma1.it

Seminar Reasoning Agents
May 26, 2022

Agenda

- Recap on Planning for Temporally Extended Goals
- Planning Domain Definition Language (PDDL)
- Compiling DFAs within PDDL
- The FOND4LTLf Tool

Planning Settings

Deterministic Domain, Reachability Goal

Non-Deterministic Domain (FOND), Reachability Goal

Deterministic Domain, LTL_f/LDL_f Goal

Non-Deterministic Domain (FOND), LTL_f/LDL_f Goal

Why temporally extended goals?

- Idea:
    - Capture a richer class of plans (more general specifications)
    - Restrict the way the planner achieves the goal
- This problem has a long history in the AI Planning community
    - Deterministic planning [BacchusKabanza98;DeGiacomoVardi99;DohertyKvarnstram01;BaierMcIlraith06;...]
    - Non-deterministic planning [Patrizietal13;Camachoetal17,18;DeGiacomoRubin18;...]
- Actually, we can use any LTL_f /LDL_f /$PLTL_f$ /$PLDL_f$ formalisms to represent planning goals

FOND planning

Nondeterministic domain (including initial state)

$\mathcal{D} = (2^{\mathcal{F}}, \mathcal{A}, s_0, \delta, \alpha)$ where:

- $\mathcal{F}$ **fluents** (atomic propositions)
- $\mathcal{A}$ **actions** (atomic symbols)
- $2^{\mathcal{F}}$ set of states
- s_0 initial state (initial assignment to fluents)
- $\alpha(s) \subseteq \mathcal{A}$ represents **action preconditions**
- $\delta(s, a, s')$ with $a \in \alpha(s)$ represents **action effects (including frame)**.

Who controls what?

Fluents controlled by **environment**

Actions controlled by **agent**

Observe: $\delta(s, a, s')$

Planning Overview

Deterministic, Reachability Goal

Goal: Propositional formula on fluents
Plan: Sequence of actions
Planning: Reach a final goal state (PSPACE-complete)

FOND, Reachability Goal

Goal: Propositional formula on fluents
Plan: strategy for agt to win the game
Planning: 2-player game (agt-env) (EXPTIME-complete)

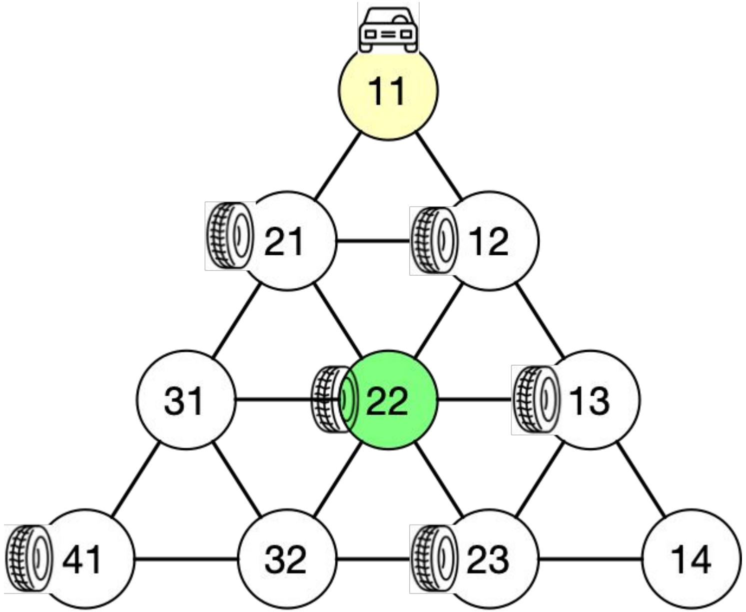
Deterministic, LTL_f/LDL_f Goal

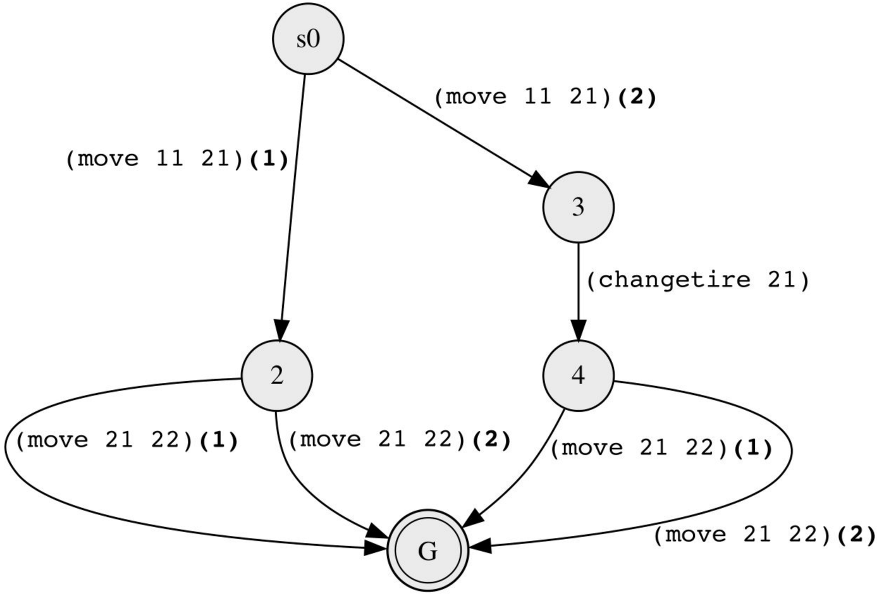
Goal: LTL_f/LDL_f formula on fluents
Plan: Sequence of actions
Planning: Find a satisfying trace for both D and goal (PSPACE-complete in D and in the goal)

FOND, LTL_f/LDL_f Goal

Goal: LTL_f/LDL_f formula on fluents
Plan: strategy for agt to win the game
Planning: 2-player game (agt-env) (EXPTIME-complete in D, 2EXPTIME-complete in goal)

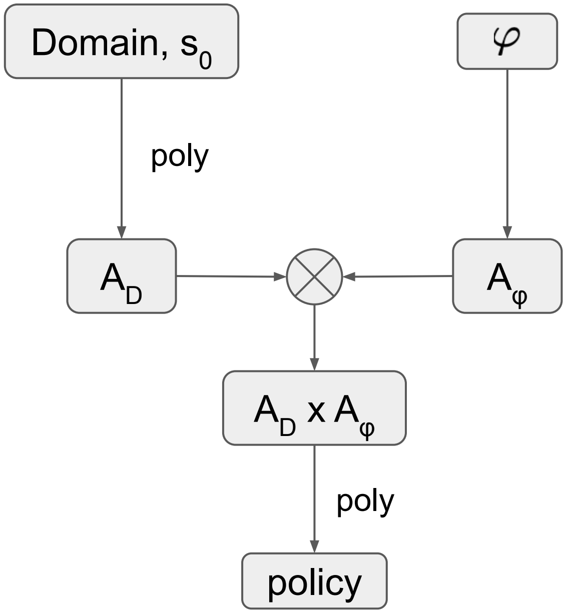
Running example - TriangleTireworld



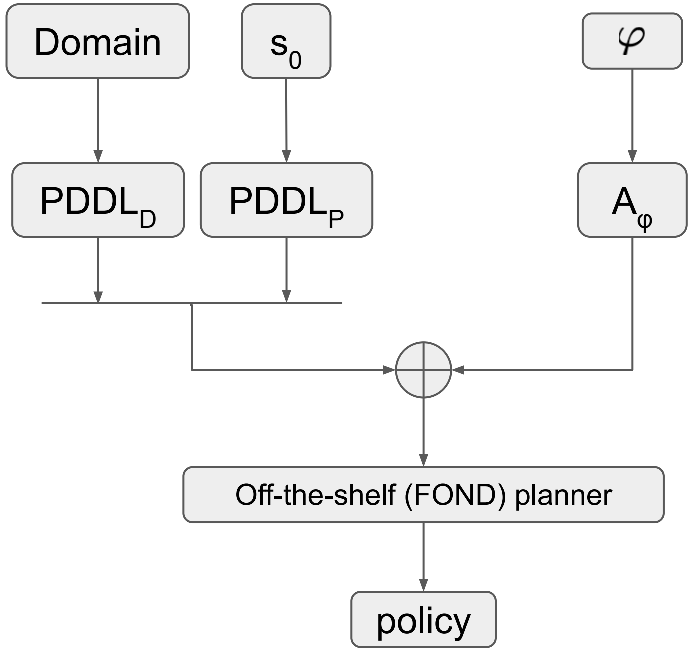


Solutions to Planning for TEGs

Automata-theoretic Techniques



Automata Encoding in PDDL



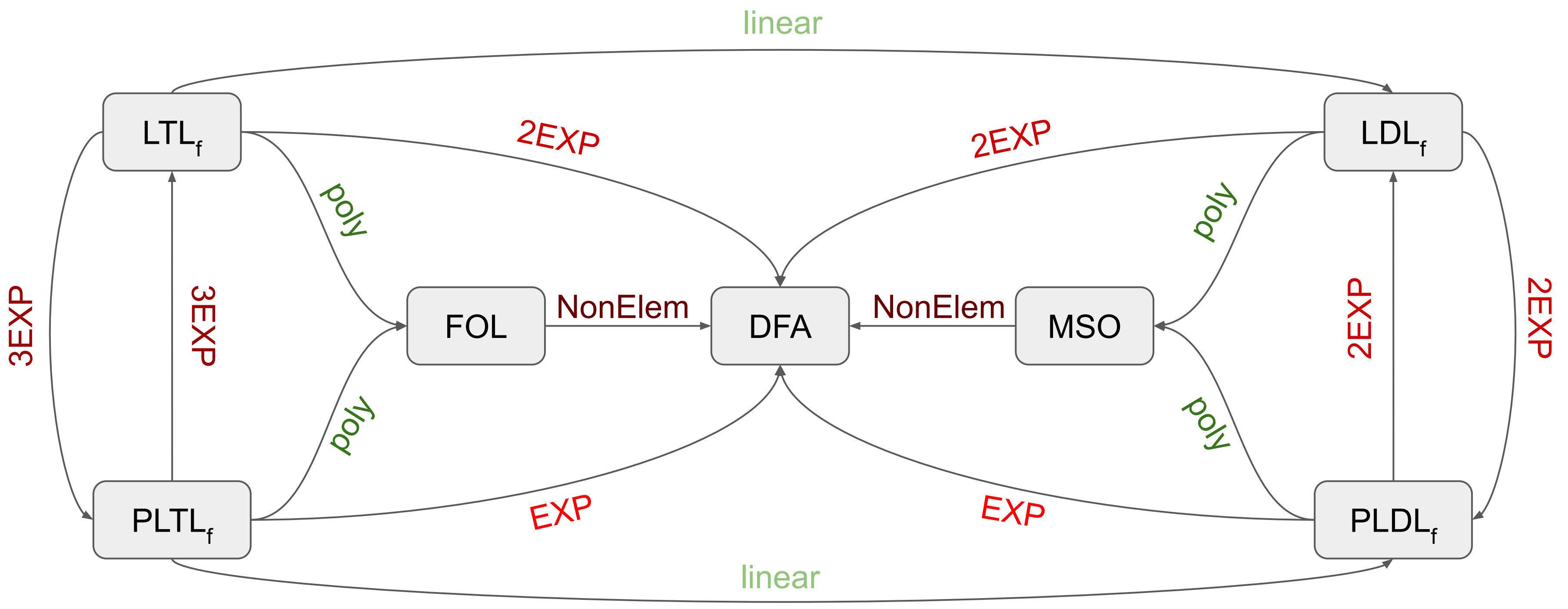
Key property

LTL_f / LDL_f formulas can be translated into *deterministic finite-state automata* (DFA)

$$\tau \models \varphi \text{ iff } \tau \in \mathcal{L}(A_\varphi)$$

where A_φ is the DFA associated to φ

Many other formalisms…



Recap on PDDL (i)

PDDL = Planning Domain Definition Language

PDDL is the de-facto standard for the specification of planning tasks

Main components of a PDDL planning tasks:

- **Objects:** Things in the world that interest us
- **Predicates:** Properties of objects that we are interested in; can be *true* or *false*.
- **Initial state:** The state of the world that we start in.
- **Goal specification:** Things that we want to be true. (classical setting)
- **Actions/Operators:** Ways of changing the state of the world.

Recap on PDDL (ii)

Planning tasks specified in PDDL are separated into two files:

1. A **domain file** for predicates and actions.
2. A **problem file** for objects, initial state and goal specification.

Note:

- Generally, PDDL domains are *independent* from PDDL problems. We can have several problems for a specific domain.
- PDDL domains are parametric. They are instantiated/grounded (becoming propositional) at planning time.

Running example PDDL domain

```
(define (domain triangle-tire)
  (:requirements :typing :strips :non-deterministic)
  (:types location)
  (:predicates
    (vehicleat ?loc - location)
    (spare-in ?loc - location)
    (road ?from - location ?to - location)
    (not-flattire)
  )

  (:action move-car
    :parameters (?from - location ?to - location)
    :precondition (and (vehicleat ?from) (road ?from ?to) (not-flattire))
    :effect (and
            (oneof
                (and (vehicleat ?to) (not (vehicleat ?from)))
                (and (vehicleat ?to) (not (vehicleat ?from))
                     (not (not-flattire)))
            )
        ))

  (:action changetire
    :parameters (?loc - location)
    :precondition (and (spare-in ?loc) (vehicleat ?loc))
    :effect (and (not (spare-in ?loc)) (not-flattire))))
```

Running example PDDL problem

```
(define (problem triangletire)
  (:domain triangletire)
  (:objects 11 12 13 14
            21 22 23
            31 32
            41 - location)
  (:init    (vehicleat 11)
            (road 11 12)(road 12 13)(road 13 14)
            (road 11 21)(road 12 22)
            (road 13 23)(road 21 12) ...

            (spare-in 12)(spare-in 13)
            (spare-in 21)(spare-in 22)(spare-in 23)
            (spare-in 41)

            (not-flattire))

  (:goal (vehicleat 14)))
```

PDDL: a note on action effects

Action effects can be more complicated than what seen so far

They can be **conditional:**

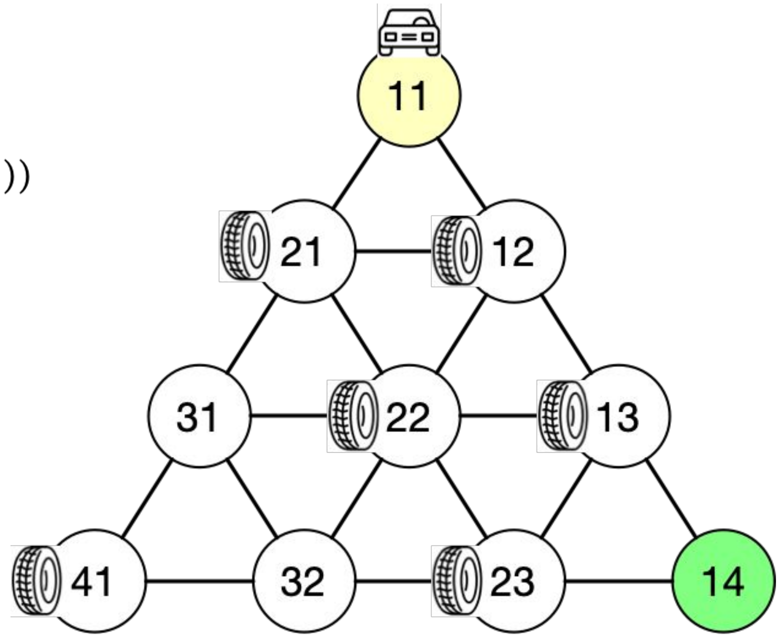
```
(when <condition>
      <effect>)
```

They can be **universally quantified**:

```
(forall (?v1 ... ?vn)
        <effect>)
```

Possible TEGs for our example

- $\Diamond(\mathsf{vehicleat}(14))$
- $\Diamond(\mathsf{vehicleat}(12) \wedge \mathcal{X}(\Diamond(\mathsf{vehicleat}(13) \wedge \mathcal{X}(\Diamond(\mathsf{vehicleat}(14))))))$
- $\mathsf{vehicleat}(14) \wedge \diamondminus(\mathsf{vehicleat}(22))$
- $\mathsf{vehicleat}(14) \wedge \ominus(\mathsf{vehicleat}(23))$
- $\mathsf{vehicleat}(14) \wedge (\neg\mathsf{vehicleat}(13)\ \mathcal{S}\ \mathsf{vehicleat}(12))$
- ...



A simple solution

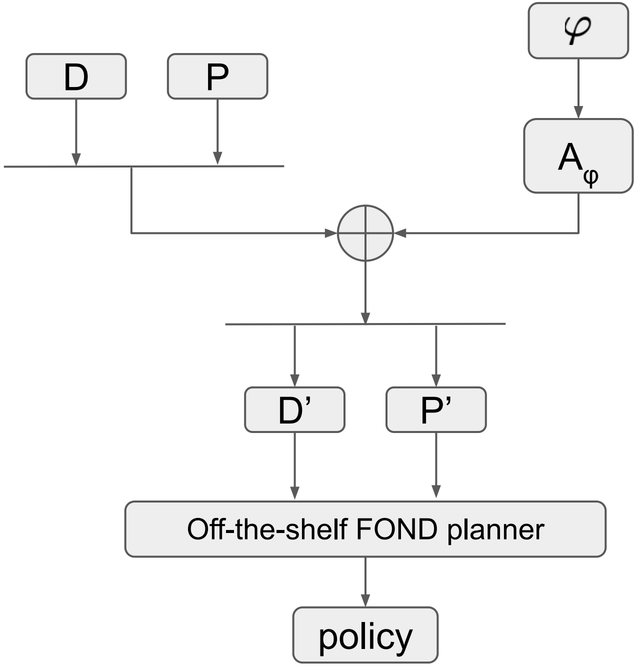
Steps:

1. Build the *parametric* DFA of A_φ (PDFA)
2. Encode dynamics of the PDFA in PDDL
3. Generate <D', P'>

This solution has been implemented:

- whitemech/fond4ltlf
- https://fond4ltlf.herokuapp.com

Automata Encoding in PDDL



1. Build the *parametric* DFA

Why? In our DFA, propositions are represented by domain fluents grounded on specific objects of interest, but in the PDDL domain this is not the case! So, we replace propositions with objects variables

How will the policy "talk" about our specific objects? We use a mapping function m^{obj} that maps objects variables into the problem instance (i.e., in P')

DEFINITION 6. *A parametric DFA (PDFA) is a tuple* $\mathcal{A}^p_\varphi = \langle \Sigma^p, Q^p, q^p_0, \delta^p, F^p \rangle$*, where:*

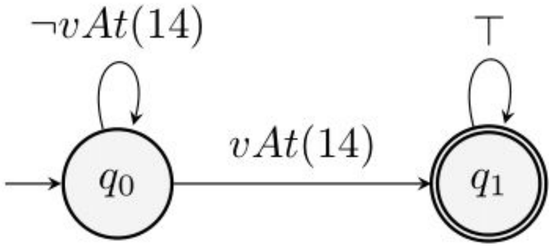
- $\Sigma^p = \{\sigma^p_0, \ldots, \sigma^p_n\} = 2^{\mathcal{F}}$ *is the alphabet of planning domain fluents;*
- Q^p *is a nonempty set of parametric states;*
- q^p_0 *is the parametric initial state;*
- $\delta^p : Q^p \times \Sigma^p \to Q^p$ *is the parametric transition function;*
- $F^p \subseteq Q^p$ *is the set of parametric final states.*

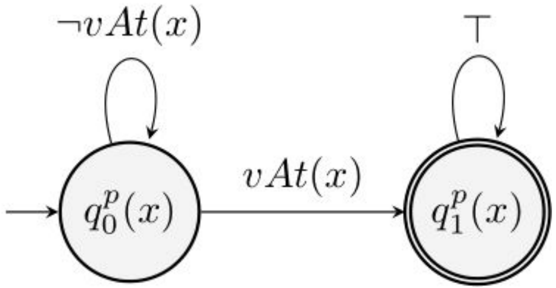
$\Sigma^p, Q^p, q^p_0, \delta^p$ *and* F^p *can be obtained by applying* m^{obj} *to all the components of the corresponding DFA.*

1. Build the *parametric* DFA (example)

Our LTL_f goal formula:

$$\Diamond \mathsf{vehicleat}(14)$$





2. Dynamics of PDFA in PDDL

New Fluents of D': $F \cup \{q \mid q \in Q^p\} \cup \{\texttt{turnDomain}\}$

Modified Agent Actions in D': for every $a \in A$

$Pre'_a = Pre_a \cup \{\texttt{turnDomain}\}$

$\mathit{Eff}'_a = \mathit{Eff}_a \cup \{\texttt{(not (turnDomain))}\}$

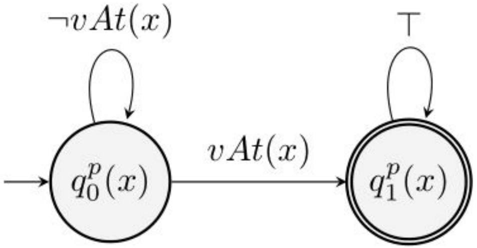
New PDFA transition function in D':

$Pre_{\texttt{transition}} = \{\texttt{(not (turnDomain))}\}$

$\mathit{Eff}_{\texttt{transition}} = \{\texttt{turnDomain}\} \cup \{\texttt{when } (q^p, \sigma^p), \texttt{then } \delta^p(q^p, \sigma^p) \cup \{\neg q \mid q \neq q^p, q \in Q^p\}\}$, for all $(q^p, \sigma^p) \in \delta^p$.

2. Dynamics of PDFA in PDDL (example)

```
(:action transition
  :parameters (?x - location)
  :precondition (not (turnDomain))
  :effect (and (when (and (q0 ?x) (not (vAt ?x)))
                     (and (q0 ?x) (not (q1 ?x)) (turnDomain))
               (when (or (and (q0 ?x) (vAt ?x)) (q1 ?x))
                     (and (q1 ?x) (not (q0 ?x)) (turnDomain))
          )
)
```



3. Generate <D', P'>

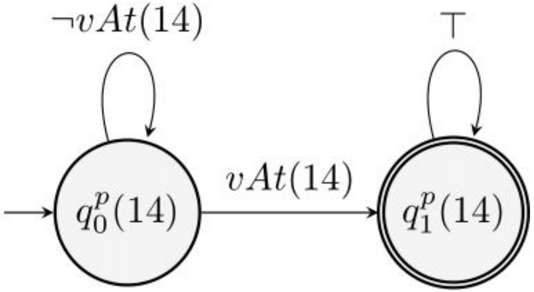
New Initial Condition in P': $s'_0 = s_0 \cup \{q_0^p\} \cup \{\texttt{turnDomain}\}$

New Goal Condition in P': $G' = \{\bigvee q_i \mid q_i \in F^p\} \cup \{\texttt{turnDomain}\}$

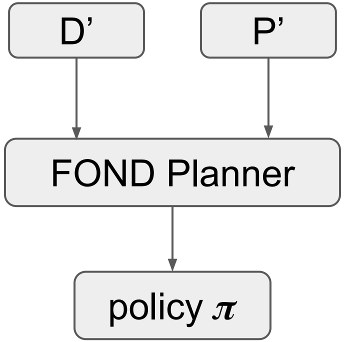
In our running example:

```
(:init (and (road 11 21) (road 11 21) ...
            (spare-in 21) (spare-in 12) ...
            (q0 14)
            (turnDomain)
       )
)
```

```
(:goal
    (and (q1 14) (turnDomain))
)
```



Planners and policy



- FOND-SAT
- MyND
- PRP
- Paladinus

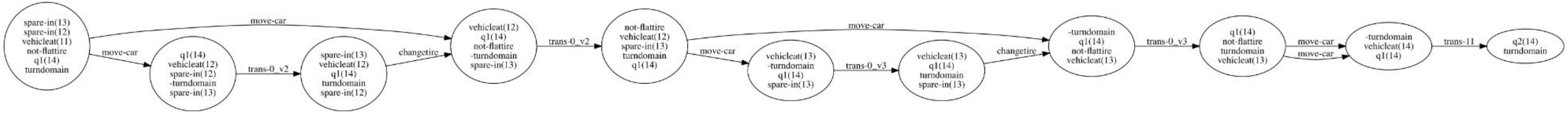
Given the policy, its executions will be of the form

$$e_i^{\pi} : [a_1, t_1, a_2, t_2, \ldots, a_n, t_n]$$

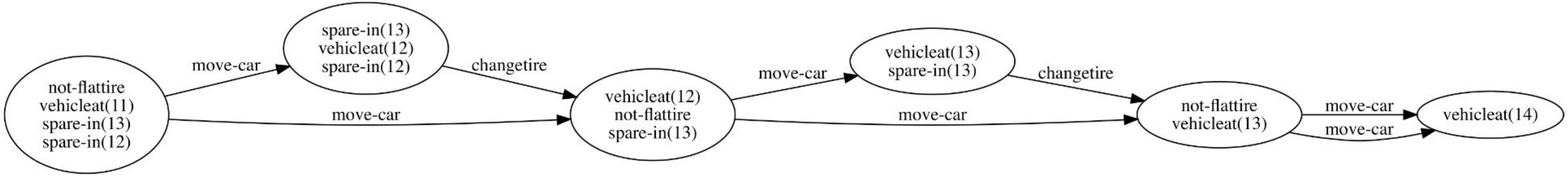
where $a_1 \ldots a_n$ *are agent actions* and $t_1 \ldots t_n$ are synchronization "transition" actions, which, at the end, can be easily removed to extract the desired execution (plan).

Results for our running example

Policy with “transitions” of the DFA



Final policy



Main Limitations

- This encoding is not very efficient
  - FOND Planning for TEGs is EXPTIME-complete in the size of the domain (*number of fluents*), 2EXPTIME-complete in the size of the LTL_f /LDL_f goal
  - Each DFA state is a new fluent in the domain, and DFAs can be very large!!!

From the planner's side:

- State-of-the-art FOND planners do not fully support conditional effects (and other PDDL features)
- Performances are not great, especially for SAT-based planners
- Some problems with forms of non-determinism that involve many misleading plans

The FOND4LTL$_f$ Tool Demo