# Planning for $\text{LTL}_f$ and $\text{LDL}_f$ Goals

Giuseppe De Giacomo

Università degli Studi di Roma "La Sapienza"
Roma, Italy



## Outline

## Outline





## $\text{LTL}_f$: LTL over finite traces

### $\text{LTL}_f$: the language

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

- $A$: atomic propositions
- $\neg\varphi$, $\varphi_1 \wedge \varphi_2$: boolean connectives
- $\bigcirc\varphi$: "next step exists and at next step (of the trace) $\varphi$ holds"
- $\varphi_1 \,\mathcal{U}\, \varphi_2$:"eventually $\varphi_2$ holds, and $\varphi_1$ holds until $\varphi_2$ does"
- $\bullet\varphi \doteq \neg\bigcirc\neg\varphi$: "if next step exists then at next step $\varphi$ holds" *(weak next)*
- $\Diamond\varphi \doteq \mathtt{true}\,\mathcal{U}\,\varphi$: "$\varphi$ will eventually hold"
- $\Box\varphi \doteq \neg\Diamond\neg\varphi$: "from current till last instant $\varphi$ will always hold"
- $Last \doteq \neg\bigcirc\mathtt{true}$: denotes last instant of trace.

### Main formal properties:

- **Expressibility:** FOL over finite sequences or Star-free RE
- **Reasoning:** satisfiability, validity, entailment PSPACE-complete
- **Model Checking:** linear on TS, PSPACE-complete on formula

## LTL$_f$: LTL over finite traces

Some interesting LTL$_f$ formulas:

| *name of template* | *LTL semantics* |
|---|---|
| *responded existence*$(A, B)$ | $\Diamond A \Rightarrow \Diamond B$ |
| *co-existence*$(A, B)$ | $\Diamond A \Leftrightarrow \Diamond B$ |
| *response*$(A, B)$ | $\Box(A \Rightarrow \Diamond B)$ |
| *precedence*$(A, B)$ | $(\neg B\ \mathcal{U} A) \vee \Box(\neg B)$ |
| *succession*$(A, B)$ | *response*$(A, B) \wedge$ *precedence*$(A, B)$ |
| *alternate response*$(A, B)$ | $\Box(A \Rightarrow \bigcirc(\neg A\ \mathcal{U} B))$ |
| *alternate precedence*$(A, B)$ | *precedence*$(A, B) \wedge$ $\Box(B \Rightarrow \bigcirc(precedence(A, B)))$ |
| *alternate succession*$(A, B)$ | *alternate response*$(A, B) \wedge$ *alternate precedence*$(A, B)$ |
| *chain response*$(A, B)$ | $\Box(A \Rightarrow \bigcirc B)$ |
| *chain precedence*$(A, B)$ | $\Box(\bigcirc B \Rightarrow A)$ |
| *chain succession*$(A, B)$ | $\Box(A \Leftrightarrow \bigcirc B)$ |

| *name of template* | *LTL semantics* |
|---|---|
| *not co-existence*$(A, B)$ | $\neg(\Diamond A \wedge \Diamond B)$ |
| *not succession*$(A, B)$ | $\Box(A \Rightarrow \neg(\Diamond B))$ |
| *not chain succession*$(A, B)$ | $\Box(A \Rightarrow \bigcirc(\neg B))$ |

| *name of template* | *LTL semantics* |
|---|---|
| *existence*$(1, A)$<br>*existence*$(2, A)$<br>...<br>*existence*$(n, A)$ | $\Diamond A$<br>$\Diamond(A \wedge \bigcirc(existence(1, A)))$<br>...<br>$\Diamond(A \wedge \bigcirc(existence(n-1, A)))$ |
| *absence*$(A)$ | $\neg existence(1, A)$ |
| *absence*$(2, A)$<br>*absence*$(3, A)$<br>...<br>*absence*$(n+1, A)$ | $\neg existence(2, A)$<br>$\neg existence(3, A)$<br>...<br>$\neg existence(n+1, A)$ |
| *init*$(A)$ | $A$ |



## LDL$_f$: LDL over finite traces

### LDL$_f$: the language

$$\varphi ::= \texttt{tt} \mid \texttt{ff} \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \langle\rho\rangle\varphi \mid [\rho]\varphi \qquad \rho ::= \phi \mid \varphi? \mid \rho_1 + \rho_2 \mid \rho_1; \rho_2 \mid \rho^*$$

- tt and ff stand for true and false
- $\phi$: propositional formula on current state/instant
- $\neg\varphi$, $\varphi_1 \wedge \varphi_2$: boolean connectives
- $\rho$ is a regular expression on propositional formulas
- $\langle\rho\rangle\varphi$: exists an "execution" of RE $\rho$ that ends with $\varphi$ holding
- $[\rho]\varphi$: all "executions" of RE $\rho$ *(along the trace!)* end with $\varphi$ holding

*In the infinite trace setting, such enhancement strongly advocated by industrial model checking (ForSpec, PSL).*

### Main formal properties:

- **Expressibility:** MSO over finite sequences: adds the power of recursion (as RE)
- **Reasoning:** satisfiability, validity, entailment PSPACE-complete
- **Model Checking:** linear on TS, PSPACE-complete on formula

## $\text{LDL}_f$: Linear Dynamic Logic on finite traces

### Example

- All coffee requests from person $p$ will eventually be served:
$$[\texttt{true}^*](request_p \supset \langle\texttt{true}^*\rangle coffee_p)$$
- Every time the robot opens door $d$ it closes it immediately after:
$$[\texttt{true}^*]([openDoor_d]closeDoor_d)$$
- Before entering restricted area $a$ the robot must have permission for $a$:
$$\langle(\neg inArea_a{}^*; getPermission_a; \neg inArea_a{}^*; inArea_a)^*; \neg inArea_a{}^*\rangle end$$

Note that the first two properties (not the third one) can be expressed also in $\text{LTL}_f$:

$$\Box(request_p \supset \Diamond coffee_p) \qquad \Box(openDoor_d \supset \bigcirc closeDoor_d)$$



## $\text{LDL}_f$: Linear Dynamic Logic on finite traces

$\text{LDL}_f$, not $\text{LTL}_f$, is able to easily express procedural constraints [BaierFritzMcIlraith07].

### Let's introduce a sort of propositional variant of GOLOG

$$\delta ::= A \mid \varphi? \mid \delta_1 + \delta_2 \mid \delta_1;\delta_2 \mid \delta^* \mid \textbf{if } \phi \textbf{ then } \delta_1 \textbf{ else } \delta_2 \mid \textbf{while } \phi \textbf{ do } \delta$$

where **if** and **while** can be seen as abbreviations for $\text{LDL}_f$ path expression, namely:

$$\textbf{if } \phi \textbf{ then } \delta_1 \textbf{else } \delta_2 \doteq (\phi?;\delta_1) + (\neg\phi?;\delta_2) \qquad \textbf{while } \phi \textbf{ do } \delta \doteq (\phi?;\delta)^*; \neg\phi?$$

### Example ($\text{LDL}_f$ procedural constraints)

- "At every point, if it is hot then, if the air-conditioning system is off, turn it on, else don't turn it off":
$$\begin{array}{l}[\texttt{true}^*]\langle \textbf{if } (hot) \textbf{ then} \\ \qquad \textbf{if } (\neg airOn) \textbf{ then } turnOnAir \\ \qquad \textbf{else } \neg turnOffAir\rangle true\end{array}$$
- "alternate till the end the following two instractions: (1) while is hot if the air-conditioning system is off turn it on, else don't turn it off; (2) do something for one step"
$$\begin{array}{l}\langle(\textbf{while } (hot) \textbf{ do} \\ \qquad \textbf{if } (\neg airOn) \textbf{ then } turnOnAir \\ \qquad \textbf{else } \neg turnOffAir; \\ \quad \texttt{true})^*\rangle end\end{array}$$

## $\text{LDL}_f$: Linear Dynamic Logic on finite traces

### Example ($\text{LDL}_f$ captures finite domain variant of GOLOG in SitCalc)

#### GOLOG – finite domain variant

$$\delta \quad ::= \quad A \mid \varphi? \mid \delta_1 + \delta_2 \mid \delta_1 ; \delta_2 \mid \delta^* \mid \pi x.\delta(x) \mid \textbf{if } \phi \textbf{ then } \delta_1 \textbf{else } \delta_2 \mid \textbf{while } \phi \textbf{do } \delta$$

- $\pi x.\delta(x)$ **stands for** $\Sigma_{o \in Obj}\ \delta(o)$
- **if** $\phi$ **then** $\delta_1$**else** $\delta_2$ stands for $(\phi?; \delta_1) + (\neg\phi?; \delta_2)$
- **while** $\phi$ **do** $\delta$ stands for $(\phi?; \delta)^* \neg\phi?$

- $\langle\delta\rangle\phi$ in $\text{LDL}_f$ captures SitCalc formula $\exists s'.Do(\delta, s, s') \wedge s \leq s' \leq last \wedge \phi(s')$.
- $[\delta]\phi$ in $\text{LDL}_f$ captures SitCalc formula $\forall s'.Do(\delta, s, s') \wedge s \leq s' \leq last \supset \phi(s')$.

*( $\phi(s)$ "uniform" in $s$.)*



## Outline

## LTL$_f$/LDL and automata

### Key point

LTL$_f$/LDL$_f$ formulas can be translated into nondeterministic finite state automata (NFA).

$$t \models \varphi \text{ iff } t \in \mathcal{L}(\mathcal{A}_\varphi)$$

where $\mathcal{A}_\varphi$ is the NFA $\varphi$ is translated into.

*We can compile reasoning into automata based procedures!*

SAPIENZA Università di Roma



## LTL$_f$/LDL$_f$ and automata

Both LTL$_f$ and LDL$_f$ formulas can be translated in exponential time to nondetermiistic automata on finite words (NFA).

### NFA $\mathcal{A}_\varphi$ associated with an LTL$_f$ formula $\varphi$ (in NNF)

#### Auxiliary rules

$$
\begin{array}{lcl}
\delta(A,\Pi) & = & \texttt{true} \text{ if } A \in \Pi \\
\delta(A,\Pi) & = & \texttt{false} \text{ if } A \not\in \Pi \\
\delta(\neg A,\Pi) & = & \texttt{false} \text{ if } A \in \Pi \\
\delta(\neg A,\Pi) & = & \texttt{true} \text{ if } A \not\in \Pi \\
\delta(\varphi_1 \wedge \varphi_2,\Pi) & = & \delta(\varphi_1,\Pi) \wedge \delta(\varphi_2,\Pi) \\
\delta(\varphi_1 \vee \varphi_2,\Pi) & = & \delta(\varphi_1,\Pi) \vee \delta(\varphi_2,\Pi) \\
\delta(\bigcirc\varphi,\Pi) & = & \begin{cases} \varphi & \text{if } \mathit{Last} \not\in \Pi \\ \texttt{false} & \text{if } \mathit{Last} \in \Pi \end{cases} \\
\delta(\Diamond\varphi,\Pi) & = & \delta(\varphi,\Pi) \ \vee\ \delta(\bigcirc\Diamond\varphi,\Pi) \\
\delta(\varphi_1 \mathcal{U} \varphi_2,\Pi) & = & \delta(\varphi_2,\Pi) \ \vee\ (\delta(\varphi_1,\Pi) \wedge \delta(\bigcirc(\varphi_1 \mathcal{U} \varphi_2),\Pi)) \\
\delta(\bullet\varphi,\Pi) & = & \begin{cases} \varphi & \text{if } \mathit{Last} \not\in \Pi \\ \texttt{true} & \text{if } \mathit{Last} \in \Pi \end{cases} \\
\delta(\Box\varphi,\Pi) & = & \delta(\varphi,\Pi) \ \wedge\ \delta(\bullet\Box\varphi,\Pi) \\
\delta(\varphi_1 \mathcal{R} \varphi_2,\Pi) & = & \delta(\varphi_2,\Pi) \ \wedge\ (\delta(\varphi_1,\Pi) \vee \delta(\bullet(\varphi_1 \mathcal{R} \varphi_2),\Pi))
\end{array}
$$

*Observe these are the rules defining the transition function of the AFW!*

#### Algorithm

**algorithm** LTL$_f$2NFA
**input** LTL$_f$ formula $\varphi$
**output** NFA $A_\varphi = (2^{\mathcal{P}}, \mathcal{S}, \{s_0\}, \varrho, \{s_f\})$
$s_0 \leftarrow \{\varphi\}$ ▷ single initial state
$s_f \leftarrow \emptyset$ ▷ single final state
$\mathcal{S} \leftarrow \{s_0, s_f\}$, $\varrho \leftarrow \emptyset$
**while** ($\mathcal{S}$ or $\varrho$ change) **do**
  **if**($q \in \mathcal{S}$ and $q' \models \bigwedge_{(\psi \in q)} \delta(\psi,\Pi)$)
    $\mathcal{S} \leftarrow \mathcal{S} \cup \{q'\}$ ▷ update set of states
    $\varrho \leftarrow \varrho \cup \{(q,\Pi,q')\}$ ▷ update transition relation

SAPIENZA Università di Roma

## $\text{LTL}_f/\text{LDL}_f$ and automata

### NFA $\mathcal{A}_\varphi$ associated with an $\text{LDL}_f$ formula $\varphi$ (in NNF)

#### Auxiliary rules

$$
\begin{array}{lcl}
\delta(\texttt{tt}, \Pi) & = & \texttt{true} \\
\delta(\texttt{ff}, \Pi) & = & \texttt{false} \\
\delta(\varphi_1 \wedge \varphi_2, \Pi) & = & \delta(\varphi_1, \Pi) \wedge \delta(\varphi_2, \Pi) \\
\delta(\varphi_1 \vee \varphi_2, \Pi) & = & \delta(\varphi_1, \Pi) \vee \delta(\varphi_2, \Pi) \\
\delta(\langle \phi \rangle \varphi, \Pi) & = & \begin{cases} \texttt{false} & \text{if } \Pi \not\models \phi \text{ or } \Pi = \epsilon \text{ (trace ended)} \\ \mathbf{e}(\varphi) & \text{o/w} \quad (\phi \text{ propositional}) \end{cases} \\
\delta(\langle \psi? \rangle \varphi, \Pi) & = & \delta(\psi, \Pi) \wedge \delta(\varphi, \Pi) \\
\delta(\langle \rho_1 + \rho_2 \rangle \varphi, \Pi) & = & \delta(\langle \rho_1 \rangle \varphi, \Pi) \vee \delta(\langle \rho_2 \rangle \varphi, \Pi) \\
\delta(\langle \rho_1 ; \rho_2 \rangle \varphi, \Pi) & = & \delta(\langle \rho_1 \rangle \langle \rho_2 \rangle \varphi, \Pi) \\
\delta(\langle \rho^* \rangle \varphi, \Pi) & = & \delta(\varphi, \Pi) \vee \delta(\langle \rho \rangle \mathbf{f}_{\langle \rho^* \rangle \varphi}, \Pi) \\
\delta([\phi] \varphi, \Pi) & = & \begin{cases} \texttt{true} & \text{if } \Pi \not\models \phi \text{ or } \Pi = \epsilon \text{ (trace ended)} \\ \mathbf{e}(\varphi) & \text{o/w} \quad (\phi \text{ propositional}) \end{cases} \\
\delta([\psi?] \varphi, \Pi) & = & \delta(nnf(\neg \psi), \Pi) \vee \delta(\varphi, \Pi) \\
\delta([\rho_1 + \rho_2] \varphi, \Pi) & = & \delta([\rho_1] \varphi, \Pi) \wedge \delta([\rho_2] \varphi, \Pi) \\
\delta([\rho_1 ; \rho_2] \varphi, \Pi) & = & \delta([\rho_1][\rho_2] \varphi, \Pi) \\
\delta([\rho^*] \varphi, \Pi) & = & \delta(\varphi, \Pi) \wedge \delta([\rho] \mathbf{t}_{[\rho^*] \varphi}, \Pi) \\
\delta(\mathbf{f}_\psi, \Pi) & = & \texttt{false} \\
\delta(\mathbf{t}_\psi, \Pi) & = & \texttt{true}
\end{array}
$$

($\mathbf{e}(\varphi)$ replaces in $\varphi$ all occurrences of $\mathbf{t}_\psi$ and $\mathbf{f}_\psi$ by $\mathbf{e}(\psi)$)

#### Algorithm

**algorithm** $\text{LDL}_f 2\text{NFA}$
**input** $\text{LDL}_f$ formula $\varphi$
**output** NFA $A_\varphi = (2^{\mathcal{P}}, \mathcal{S}, \{s_0\}, \varrho, \{s_f\})$
$s_0 \leftarrow \{\varphi\}$ ▷ single initial state
$s_f \leftarrow \emptyset$ ▷ single final state
$\mathcal{S} \leftarrow \{s_0, s_f\}$, $\varrho \leftarrow \emptyset$
**while** ($\mathcal{S}$ or $\varrho$ change) **do**

  **if**($q \in \mathcal{S}$ and $q' \models \bigwedge_{(\psi \in q)} \delta(\psi, \Pi)$)
    $\mathcal{S} \leftarrow \mathcal{S} \cup \{q'\}$ ▷ update set of states
    $\varrho \leftarrow \varrho \cup \{(q, \Pi, q')\}$ ▷ update transition relation





## $\text{LTL}_f/\text{LDL}_f$ reasoning

### $\text{LTL}_f/\text{LDL}_f$ satisfiability ($\varphi$ SAT)

1: Given $\text{LTL}_f/\text{LDL}_f$ formula $\varphi$
2: Compute NFA for $\varphi$ *(exponential)*
3: Check NFA for nonemptiness *(NLOGSPACE)*
4: Return result of check

### $\text{LTL}_f/\text{LDL}_f$ validity ($\varphi$ VAL)

1: Given $\text{LTL}_f/\text{LDL}_f$ formula $\varphi$
2: Compute NFA for $\neg\varphi$ *(exponential)*
3: Check NFA for nonemptiness *(NLOGSPACE)*
4: Return complemented result of check

### $\text{LTL}_f/\text{LDL}_f$ logical implication ($\Gamma \models \varphi$)

1: Given $\text{LTL}_f/\text{LDL}_f$ formulas $\Gamma$ and $\varphi$
2: Compute NFA for $\Gamma \wedge \neg\varphi$ *(exponential)*
3: Check NFA for nonemptiness *(NLOGSPACE)*
4: Return complemented result of check

**Thm:**[IJCAI13] All above reasoning tasks are PSPACE-complete. *(As for infinite traces.)*
*(Construction of NFA can be done while checking nonemptiness.)*

### Relationship to Classical Planning

Let $\Psi_{domain}$ describe action domain ($\text{LTL}_f$ formula), $\phi_{init}$ initial state (prop. formula), and $G$ goal (prop. formula).
Classical planning amounts to $\text{LTL}_f$ satisfiability of:

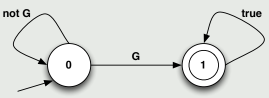
$$\phi_{init} \wedge \Psi_{domain} \wedge \Diamond G$$

*Complexity: PSPACE-complete.*

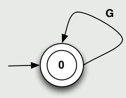
## Automata for some $\text{LTL}_f/\text{LDL}_f$ formulas

### Example (Automata for some $\text{LTL}_f/\text{LDL}_f$ formulas)

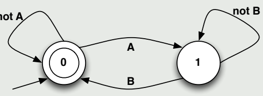
$\Diamond G$



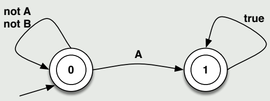

$\Box G$




$\Box(A \supset \bigcirc\Diamond B)$

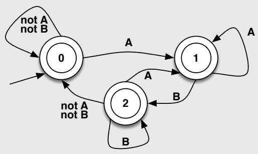



$\neg B \,\mathcal{U}\, A \vee \Box\neg B$ *"$A$ before $B$"*




$\langle(\neg B^*; A; \neg B^*; B; B)^*; \neg B^*\rangle end$

*"each time new $A$ before $B$"* *($A$ and $B$ not true simultaneously)*




*(online software for LTLf2DFA: `http://ltlf2dfa.diag.uniroma1.it`)*
*(online software for LDLf2DFA: `https://flloat.herokuapp.com`)*



## Outline

## Planning in deterministic domain

### Deterministic 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)**.

### Traces

A **trace** for $\mathcal{D}$ is a finite sequence:

$$s_0, a_1, s_1, \cdots, a_n, s_n$$

where $s_0$ is the initial state, and $a_i \in \alpha(s_i)$ and $s_{i+1} = \delta(s_i, a_{i+1})$ for each $i$.

### Goals, planning, and plans

**Goal** = propositional formula $G$ on fluents
**Planning** = find a trace $s_0, a_1, s_1, \cdots, a_n, s_n$ such that $s_n \models G$. (PSPACE-complete)
**Plan** = project traces on actions, i.e., return $a_1, \cdots, a_n$.



## Deterministic planning domains as automata

Let's transform the planning domain $\mathcal{D} = (2^{\mathcal{F}}, \mathcal{A}, s_0, \delta, \alpha)$ into a DFA recognizing all its traces.

### DFA $A_{\mathcal{D}}$ for $\mathcal{D}$

$A_{\mathcal{D}} = (2^{\mathcal{F} \cup \mathcal{A}}, (2^{\mathcal{F}} \cup \{s_{init}\}), s_{init}, \varrho, F)$ where:

- $2^{\mathcal{F} \cup \mathcal{A}}$ alphabet (actions $\mathcal{A}$ include dummy $start$ action)
- $2^{\mathcal{F}} \cup \{s_{init}\}$ set of states
- $s_{init}$ dummy initial state
- $F = 2^{\mathcal{F}}$ (all states of the domain are final)
- $\rho(s, [a, s']) = s'$ **with** $a \in \alpha(s)$**, and** $\delta(s, a) = s'$
  $\rho(s_{init}, [start, s_0]) = s_0$

*(notation: $[a, s']$ stands for $\{a\} \cup s'$)*

### Traces

Each trace $s_0, a_1, s_1, \cdots, a_n, s_n$ of the domain $\mathcal{D}$ becomes a finite sequence:

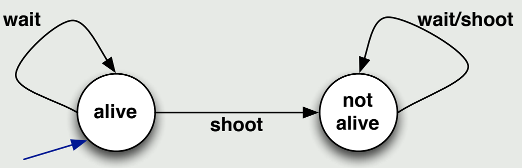
$$[start, s_0], [a_1, s_1], \cdots, [a_n, s_n]$$

recognized by the DFA $A_{\mathcal{D}}$.

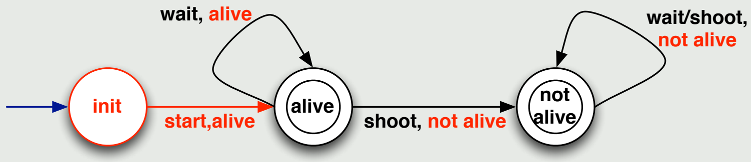
## Deterministic planning domains as automata

### Example (Simplified Yale shooting domain)

- Domain $\mathcal{D}$:




- DFA $A_{\mathcal{D}}$:




SAPIENZA
UNIVERSITÀ DI ROMA



## Deterministic planning domains as automata

### Planning in deterministic domains

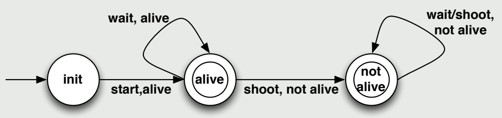
Planning = find a trace of DFA $A_{\mathcal{D}}$ for deterministic domain $\mathcal{D}$ such that is also a trace for the DFA for $\Diamond G$ where $G$ is the goal. That is:

*CHECK for nonemptiness $A_{\mathcal{D}} \cap A_{\Diamond G}$: extract plan from witness.*

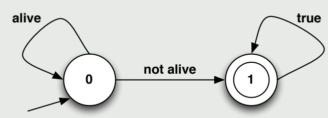
*(Computable on-the-fly, PSPACE in $\mathcal{D}$, constant in $G$. i.e., optimal)*

### Example (Simplified Yale shooting domain)

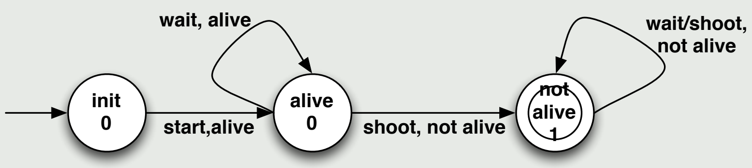
$A_{\mathcal{D}}$




$A_{\Diamond \neg alive}$




$A_{\mathcal{D}} \cap A_{\Diamond G}$:



## Generalization: planning for $\text{LTL}_f/\text{LDL}_f$ goals in deterministic domains

### Planning in deterministic domains for $\text{LTL}_f/\text{LDL}_f$ goals

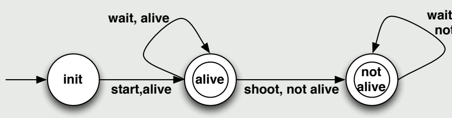
Planning = find a trace of DFA $A_\mathcal{D}$ for deterministic domain $\mathcal{D}$ such that is also a accepted by NFA $A_\varphi$ for the $\text{LTL}_f/\text{LDL}_f$ formula $\varphi$. That is:

> *CHECK for nonemptiness $A_\mathcal{D} \cap A_\varphi$: extract plan from witness.*

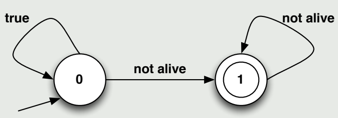
*(Computable on-the-fly, PSPACE in $\mathcal{D}$, PSPACE also in $\varphi$ i.e., optimal)*
*(We can use NFA directly since we are checking for **existence** of a trace satisfying $\varphi$)*

### Example (Simplified Yale shooting domain)

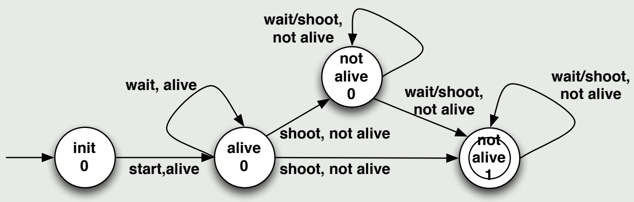
$A_\mathcal{D}$




$A_{\Diamond\Box\neg alive}$




$A_\mathcal{D} \cap A_{\Diamond\Box\neg alive}$:






## Generalization: planning for $\text{LTL}_f/\text{LDL}_f$ goals in deterministic domains

### Planning for $\text{LTL}_f/\text{LDL}_f$ goals

#### Algorithm: Planning for $\text{LDL}_f/\text{LTL}_f$ goals

1: Given $\text{LTL}_f/\text{LDL}_f$ domain $\mathcal{D}$ and goal $\varphi$
2: Compute corresponding NFA (exponential)
3: Compute intersection with DFA of $\mathcal{D}$ (polynomial)
5: Check nonemptiness of resulting NFA (NLOGSPACE)
6: Return plan

#### Theorem

*Planning for $\text{LTL}_f/\text{LDL}_f$ goals is:*

- *PSPACE-complete in the domain;*
- *PSPACE-complete in the goal.*

## Outline



SAPIENZA UNIVERSITÀ DI ROMA



## $\mathrm{FOND}_{sp}$: strong planning in nondeterministic domains

### 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:* $\delta(s, a, s')$

### Goals, planning, and plans

**Goal** = propositional formula $G$ on fluents

**Planning** = **game** between two players:

> **agent** *tries to force eventually reaching* $G$ *no matter how other* **environment** *behave.*

**Plan** = **strategy** to **win** the game.

*($\mathrm{FOND}_{sp}$ is EXPTIME-complete)*

## Nondeterministic domains as automata

Let's transform the nondeterministic domain $\mathcal{D} = (2^{\mathcal{F}}, \mathcal{A}, s_0, \delta, \alpha)$ into an automaton recognizing all its traces.

### Automaton $A_D$ for $\mathcal{D}$ is a DFA!!!

$A_{\mathcal{D}} = (2^{\mathcal{F}\cup\mathcal{A}}, (2^{\mathcal{F}} \cup \{s_{init}\}), s_{init}, \varrho, F)$ where:

- $2^{\mathcal{F}\cup\mathcal{A}}$ alphabet (actions $\mathcal{A}$ include dummy $start$ action)
- $2^{\mathcal{F}} \cup \{s_{init}\}$ set of states
- $s_{init}$ dummy initial state
- $F = 2^{\mathcal{F}}$ (all states of the domain are final)
- $\rho(s, [a, s']) = s'$ **with** $a \in \alpha(s)$, **and** $\delta(s, a, s')$ $\qquad \rho(s_{init}, [start, s_0]) = s_0$

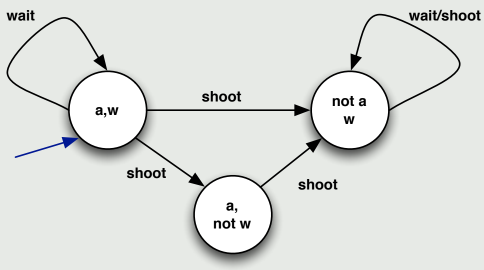
*(notation: $[a, s']$ stands for $\{a\} \cup s'$)*



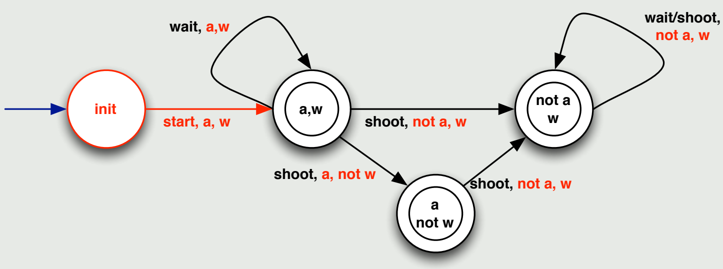
## Nondeterministic domains as automata

### Example (Simplified Yale shooting domain variant)

- Domain $\mathcal{D}$:




- DFA $A_{\mathcal{D}}$:



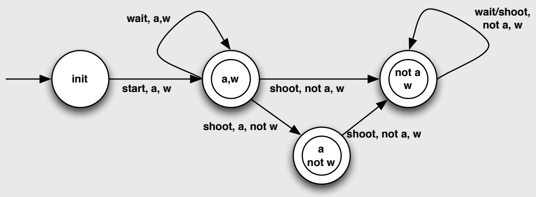
## Nondeterministic domains as automata

### $\mathrm{FOND}_{sp}$: strong planning in nondeterministic domains

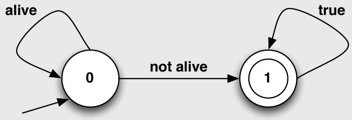
- Set the **arena** formed by all traces that satisfy both the DFA $A_\mathcal{D}$ for $\mathcal{D}$ and the DFA for $\Diamond G$ where $G$ is the goal.
- Compute a **winning strategy**. *(EXPTIME-complete in $\mathcal{D}$, constant in $G$)*

### Example (Simplified Yale shooting domain)

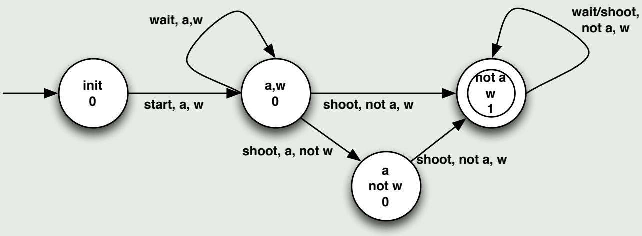
$A_\mathcal{D}$




$A_{\Diamond\neg a}$




$A_\mathcal{D} \cap A_{\Diamond\neg a}$:




strategy

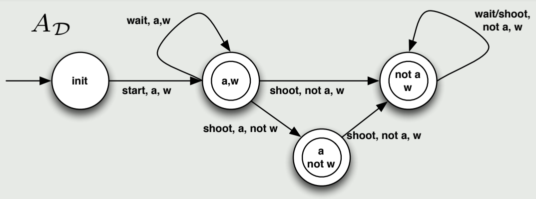
| | | |
|---|---|---|
| $init, 0$ | $\rightarrow$ | $start$ |
| $a, w, 0$ | $\rightarrow$ | $shoot$ |
| $a, \neg w, 0$ | $\rightarrow$ | $shoot$ |
| $\neg a, w, 1$ | $\rightarrow$ | win! |



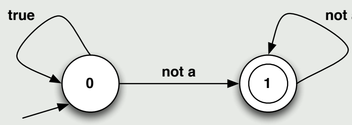
## Generalization: $\mathrm{FOND}_{sp}$ for $\mathrm{LTL}_f/\mathrm{LDL}_f$ goals

### Example (Simplified Yale shooting domain)

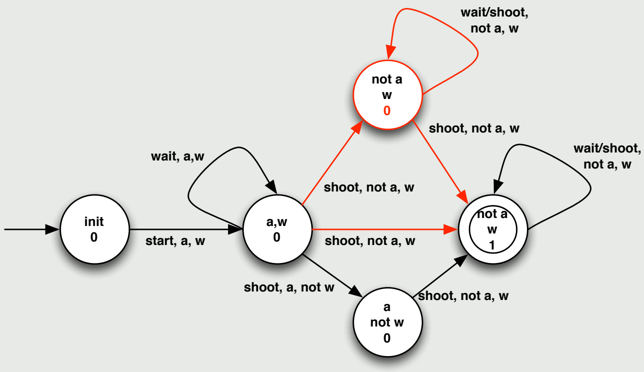
$A_\mathcal{D}$




$A_{\Diamond\Box\neg a}$




$A_\mathcal{D} \cap A_{\Diamond\Box\neg a}$:




### Can we use directly NFA's?

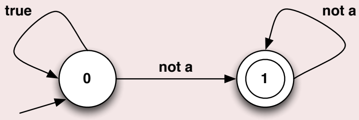
**No**, because of a basic mismatch

- NFA have perfect **foresight**, or **clairvoyance** *(angelic nondeterminism)*
- Strategies must be runnable: **depend only on past**, not future *(devilish nondeterminism)*

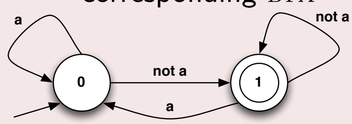
## Generalization: $\text{FOND}_{sp}$ for $\text{LTL}_f/\text{LDL}_f$ goals

### We need first to determinize the NFA for $\text{LTL}_f/\text{LDL}_f$ formula

NFA for $\Diamond\Box\neg a$



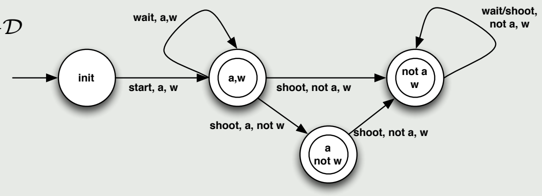

corresponding DFA



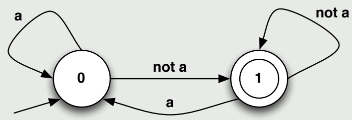

*(DFA can be exponential in NFA in general)*

### Example (Simplified Yale shooting domain)

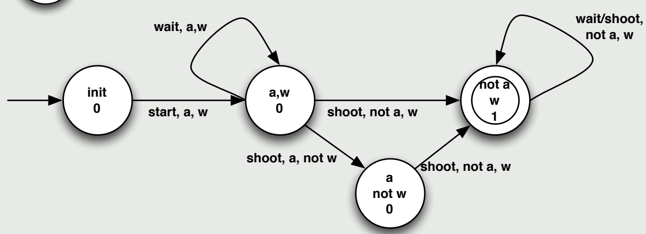
$A_{\mathcal{D}}$




$A_{\Diamond\Box\neg a}$




$A_{\mathcal{D}} \cap A_{\Diamond\Box\neg a}$:




strategy

| | | |
|---|---|---|
| $init, 0$ | $\rightarrow$ | $start$ |
| $a, w, 0$ | $\rightarrow$ | $shoot$ |
| $a, \neg w, 0$ | $\rightarrow$ | $shoot$ |
| $\neg a, w, 1$ | $\rightarrow$ | win! |



## Generalization: DFA Games

### DFA games

A DFA **game** $\mathcal{G} = (2^{\mathcal{F}\cup\mathcal{A}}, S, s_{init}, \varrho, F)$, is such that:

- $\mathcal{F}$ controlled by environment; $\mathcal{A}$ controlled by agent;
- $2^{\mathcal{F}\cup\mathcal{A}}$, alphabet of game;
- $S$, states of game;
- $s_{init}$, initial state of game;
- $\varrho : S \times 2^{\mathcal{F}\cup\mathcal{A}} \rightarrow S$, transition function of the game: given current state $s$ and a choice of action $a$ and resulting fluents values $E$ the resulting state of game is $\varrho(s, [a, E]) = s'$;
- $F$, final states of game, where game can be considered terminated.

### Winning Strategy:

- A play is **winning** for the agent if such a play leads from the initial to a final state.
- A **strategy** for the agent is a function $f : (2^{\mathcal{F}})^* \rightarrow \mathcal{A}$ that, given a **history of choices from the environment**, decides which action $\mathcal{A}$ to do next.
- A **winning strategy** is a strategy $f : (2^{\mathcal{F}})^* \rightarrow \mathcal{A}$ such that for all traces $\pi$ with $a_i = f(\pi_{\mathcal{F}}|_i)$ we have that $\pi$ leads to a final state of $\mathcal{G}$.

## Generalization: DFA Games

### Winning condition for DFA games

Let

$$PreC(\mathcal{S}) = \{s \in S \mid \exists a \in \mathcal{A}. \forall E \in 2^{\mathcal{F}}. \varrho(s, [a, E]) \in \mathcal{S}\}$$

Compute the set $Win$ of winning states of a DFA game $\mathcal{G}$, i.e., states from which the agent can win the game $\mathcal{G}$, by **least-fixpoint**:

- $Win_0 = F$ (the final states of $\mathcal{G}$)
- $Win_{i+1} = Win_i \cup PreC(Win_i)$
- $Win = \bigcup_i Win_i$

*(Computing $Win$ is linear in the number of states in $\mathcal{G}$)*

### Computing the winning strategy

Let's define $\omega : S \to 2^{\mathcal{A}}$ as:

$$\omega(s) = \{a \mid \text{ if } s \in Win_{i+1} - Win_i \text{ then } \forall E. \varrho(s, [a, E]) \in Win_i\}$$

- **Every way** of restricting $\omega(s)$ to return only one action (chosen arbitrarily) gives a **winning strategy** for $\mathcal{G}$.
- Note $s$ **is a state of the game**! not of the domain only!
  To phrase $\omega$ wrt the domain only, we need to return a **stateful transducer** with transitions from the game.



## Generalization: FOND$_{sp}$ for LTL$_f$/LDL$_f$ goals

### FOND$_{sp}$ for LTL$_f$/LDL$_f$ goals

#### Algorithm: FOND$_{sp}$ for LDL$_f$/LTL$_f$ goals

```
Given LTLf/LDLf domain D and goal φ
    Compute NFA for φ (exponential)
    Determinize NFA to DFA (exponential)
    Compute intersection with DFA of D (polynomial)
    Synthesize winning strategy for DFA game (linear)
Return strategy
```

### Theorem

*FOND$_{sp}$ for LTL$_f$/LDL$_f$ goals is:*

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

## Outline





## $\text{FOND}_{sc}$: strong cyclic planning in nondeterministic fair domains

### 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**, though under **stochastic fairness**:

*(i.e., all effects will happen with non-zero (unknown) probability)*

Actions controlled by **agent**

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

### Goals, planning, and plans

**Goal** = propositional formula $G$ on fluents

**Planning** = **agent**, in spite of the **environment**, stays in an area from where is possible to reach $G$

*(with the cooperation of **environment**! it is not a pure adversarial game!)*

**Plan** = **strategy** to stay within the good area.

*($\text{FOND}_{sc}$ is EXPTIME-complete)*

# Nondeterministic fair domains as automata

We transform the nondeterministic fair domain $\mathcal{D} = (2^{\mathcal{F}}, \mathcal{A}, s_0, \delta, \alpha)$ into an automaton recognizing all its traces exactly as we did for nondeterministic (non-fair) domains. Fairness play no role!

## Automaton $A_D$ for $\mathcal{D}$ is a DFA!!!

$A_{\mathcal{D}} = (2^{\mathcal{F} \cup \mathcal{A}}, (2^{\mathcal{F}} \cup \{s_{init}\}), s_{init}, \varrho, F)$ where:

- $2^{\mathcal{F} \cup \mathcal{A}}$ alphabet (actions $\mathcal{A}$ include dummy $start$ action)
- $2^{\mathcal{F}} \cup \{s_{init}\}$ set of states
- $s_{init}$ dummy initial state
- $F = 2^{\mathcal{F}}$ (all states of the domain are final)
- $\rho(s, [a, s']) = s'$ **with** $a \in \alpha(s)$, **and** $\delta(s, a, s')$ $\quad \rho(s_{init}, [start, s_0]) = s_0$

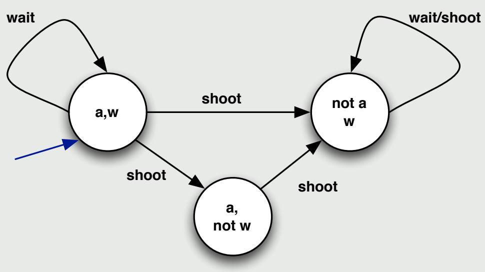
*(notation: $[a, s']$ stands for $\{a\} \cup s'$)*



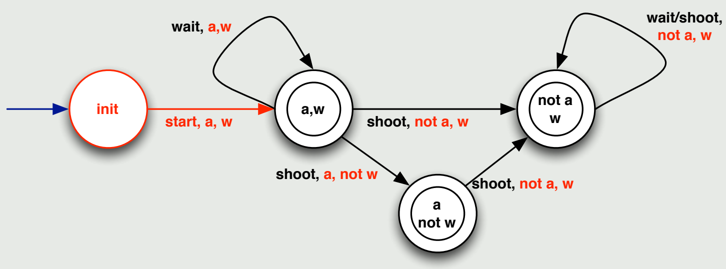
# Nondeterministic fair domains as automata

## Example (Simplified Yale shooting domain variant)

- Domain $\mathcal{D}$:




- DFA $A_{\mathcal{D}}$:



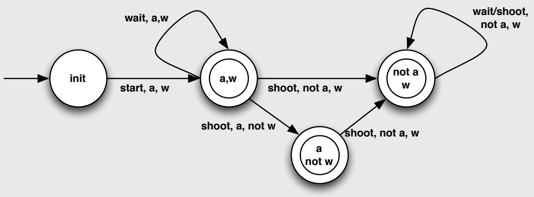
## Nondeterministic fair domains as automata

### $\text{FOND}_{sc}$: strong cyclic planning in nondeterministic domains

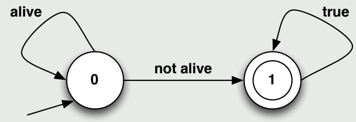
- Set the **arena** formed by all traces that satisfy both the DFA $A_{\mathcal{D}}$ for $\mathcal{D}$ and the DFA for $\Diamond G$ where $G$ is the goal.
- Compute a **winning strategy**. *(EXPTIME-complete in $\mathcal{D}$, constant in $G$)*

### Example (Simplified Yale shooting domain)

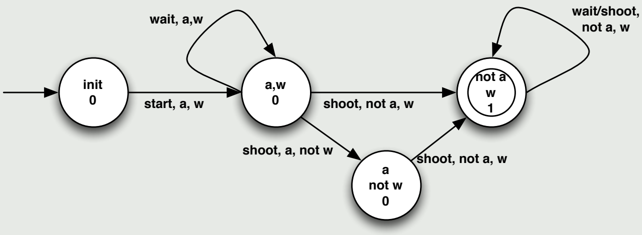
$A_{\mathcal{D}}$




$A_{\Diamond\neg a}$




$A_{\mathcal{D}} \cap A_{\Diamond\neg a}$:




strategy

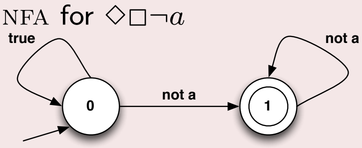
| | | |
|---|---|---|
| $init, 0$ | $\rightarrow$ | $start$ |
| $a, w, 0$ | $\rightarrow$ | $shoot$ |
| $a, \neg w, 0$ | $\rightarrow$ | $shoot$ |
| $\neg a, w, 1$ | $\rightarrow$ | win! |



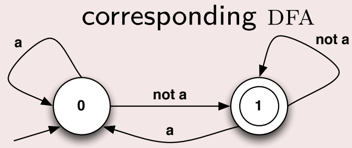
## Generalization: $\text{FOND}_{sc}$ for $\text{LTL}_f/\text{LDL}_f$ goals

### We need first to determinize the NFA for $\text{LTL}_f/\text{LDL}_f$ formula

NFA for $\Diamond\Box\neg a$



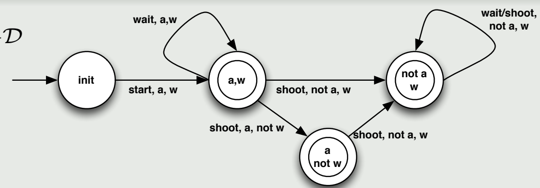

corresponding DFA



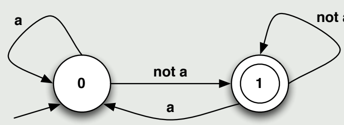

*(DFA can be exponential in NFA in general)*

### Example (Simplified Yale shooting domain)

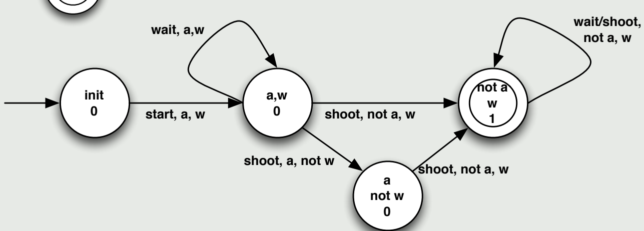
$A_{\mathcal{D}}$




$A_{\Diamond\Box\neg a}$




$A_{\mathcal{D}} \cap A_{\Diamond\Box\neg a}$:




strategy

| | | |
|---|---|---|
| $init, 0$ | $\rightarrow$ | $start$ |
| $a, w, 0$ | $\rightarrow$ | $shoot$ |
| $a, \neg w, 0$ | $\rightarrow$ | $shoot$ |
| $\neg a, w, 1$ | $\rightarrow$ | win! |

## Generalization: Fair DFA Games

Because of stochastic fairness we need to change the game we play over the DFA!

### Fair DFA games

A **fair DFA game** $\mathcal{G} = (2^{\mathcal{F}\cup\mathcal{A}}, S, s_{init}, \varrho, F)$, is such that:

- $\mathcal{F}$ controlled by environment; $\mathcal{A}$ controlled by agent;
- $2^{\mathcal{F}\cup\mathcal{A}}$, alphabet of game;
- $S$, states of game;
- $s_{init}$, initial state of game;
- $\varrho : S \times 2^{\mathcal{F}\cup\mathcal{A}} \to S$, transition relation of the game: given current state $s$ and a choice of action $a$ and resulting fluents values $E$ the resulting state of game is $s' = \varrho(s, [a, E])$;
- $F$, final states of game, where game can be considered terminated.

### Winning Strategy:

- A play is **winning** for the agent if from the initial the agent can force to remain in a safe area from which is possible to cooperatively reach the final state.
- A **strategy** for the agent is a function $f : (2^{\mathcal{F}})^* \to \mathcal{A}$ that, given a **history of choices from the environment**, decides which action $\mathcal{A}$ to do next.
- A **winning strategy** is a strategy $f : (2^{\mathcal{F}})^* \to \mathcal{A}$ such that all traces $\pi$ with $a_i = f(\pi_{\mathcal{F}}|_i)$ are winning for the agent.



## Generalization: Fair DFA Games

### Existential and universal preimages wrt the environament

$$
\begin{array}{rcl}
PreE(a, \mathcal{S}) & = & \{s \in S \mid a \in \mathcal{A}. \exists E \in 2^{\mathcal{F}}. \varrho(s, [a, E]) \in \mathcal{S}\} \\
PreA(a, \mathcal{S}) & = & \{s \in S \mid a \in \mathcal{A}. \forall E \in 2^{\mathcal{F}}. \varrho(s, [a, E]) \in \mathcal{S}\}
\end{array}
$$

### Agent forces that always agent reach a final state if environment cooperates

The winning condition of the game is defined by two nested fixpoints, a **greatest-fixpoint** (for safety) and **least fixpoint** (for reachability):

$$Safe = \nu X.\mu Y.F \cup \bigcup_{a\in\mathcal{A}} (PreA(a, X) \cap PreE(a, Y))$$

This gives rise to the following nested fixpoint computation:

- $X_0 = S$ (all states of $\mathcal{G}$)
- $X_{i+1} = Y_{i+1} = \mu Y.F \cup \bigcup_{a\in\mathcal{A}} (PreA(a, X_i) \cap PreE(a, Y))$
- $Safe = \bigcap_i X_i$

where $\mu Y.F \cup \bigcup_{a\in\mathcal{A}} (PreA(a, X_i) \cap PreE(a, Y))$ is computed as

- $Y_{i,0} = F$ (the final states of $\mathcal{G}$)
- $Y_{i,j+1} = F \cup \bigcup_{a\in\mathcal{A}} (PreA(a, X_i) \cap PreE(a, Y_{i,j}))$
- $Y_i = \bigcup_i Y_{i,j}$

*(Computing each $Y_i$ is linear in the number of states in $\mathcal{G}$, hence computing $Safe$ is quadratic.)*

## Generalization: Fair DFA Games

### Computing the winning strategy

We can stratify $Safe$ according to when a state enters the least fixpoint:

- $Reach_1 = F$,
- $Reach_{j+1} = Reach_j \cup PreA(a, Safe) \cap PreE(a, Reach_j)$.

Note that $Safe = \cup_{j \leq |S|} Reach_j$.
Let's define $\omega : S \to 2^{\mathcal{A}}$ as:

$$\omega(s) = \{a \mid \text{ if } s \in Reach_{j+1} - Reach_j \text{ then } \exists E.\varrho(s, [a, E]) \in Reach_j\}$$

- **Every way** of restricting $\omega(s)$ to return only one action (chosen arbitrarily) gives a **winning strategy** for $\mathcal{G}$.
- Note **$s$ is a state of the game**! not of the domain only!
  To phrase $\omega$ wrt the domain only, we need to return a **stateful transducer** with transitions from the game.



## Generalization: FOND$_{sc}$ for LTL$_f$/LDL$_f$ goals

### FOND$_{sc}$ for LTL$_f$/LDL$_f$ goals

#### Algorithm: FOND$_{sc}$ for LDL$_f$/LTL$_f$ goals

1: Given LTL$_f$/LDL$_f$ domain $\mathcal{D}$ and goal $\varphi$
2: Compute NFA for $\varphi$ (exponential)
3: Determinize NFA to DFA (exponential)
4: Compute intersection with DFA of $\mathcal{D}$ (polynomial)
5: Synthesize winning strategy for resulting fairDFA game (quadratic)
6: Return strategy

### Theorem

*FOND$_{sc}$ for LTL$_f$/LDL$_f$ goals is:*

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

## Outline





## $\text{POND}_{sp}$: strong planning in nondeter. domain with partial observability

### Nondeterministic partially observable domain (including initial state)

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

- $\mathcal{F}$ **fluents** (atomic propositions)
- $Obs \subseteq \mathcal{F}$ **observable fluents**
- $\mathcal{A}$ **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)**
- $obs(s) \subseteq Obs$ project $s$ on the **observable fluents only**.

### Goals, planning, and plans

**Goal** = propositional formula $G$ on fluents
**Planning** = **game** between two players:

> **agent** *tries to force eventually reaching* $G$ *no matter how other* **environment** *behave and in spite of seeing only observable fluents.*

**Plan** = **strategy** (depending on observable fluens only) to **win** the game.

> *Strategies* $f : (2^{Obs})^* \to \mathcal{A}$ *such that for all generated traces* $\pi$ *with* $E_i$ *compatible with observations* $O_i$ *and* $a_i = f(\pi_{Obs}|_i)$, *we have that* $\pi$ *satisfies eventually* $\mathcal{G}$.

*(POND, strong plans, 2-EXPTIME-complete)*

## Does Belief-State Construction for $\text{POND}_{sp}$ work for $\text{LTL}_f/\text{LDL}_f$? Yes ...

### Belief-states DFA game

Given a DFA game $\mathcal{A} = (2^{\mathcal{F}\cup\mathcal{A}}, S, s_{init}, \varrho, F)$ the associated **belief-states DFA game** is the following DFA game: $\mathcal{G}_A^{Obs} = (2^{\mathcal{F}\cup\mathcal{A}}, \mathcal{B}, B_{init}, \partial, \mathcal{F})$, where:

- $2^{\mathcal{F}\cup\mathcal{A}}$ is the alphabet which is the same of the original game;
- $\mathcal{B} = 2^S$ are the **belief states**, corresponding to sets of the states of the original game;
- $B_{init} = \{s_{init}\}$ is the initial belief-state, formed by initial state of the original game;
- $\partial : \mathcal{B} \times 2^{\mathcal{F}\cup\mathcal{A}} \to \mathcal{B}$ is the transition function: given the current belief-state $B$ and a choice of action $a$ and propositions $E$, respectively for the agent and the enviroment:
$$\partial(B, [a, E]) = \{s' \mid \exists s, E'.s \in B \wedge obs(E')=obs(E) \wedge \delta(s, [a, E]) = s'\}$$
- $\mathcal{F} = 2^F$ are the final belief-states, formed only by final states of the original game.

*NB: **belief-states DFA game** $\mathcal{G}_A^{Obs}$ is itself a DFA game (with full observability) over the $\mathcal{F}$ and $\mathcal{A}$.*



## ... Belief-State Construction is 3EXPTIME in the goal!

### Belief-State Algorithm for $\text{POND}_{sp}$ for $\text{LDL}_f/\text{LTL}_f$ goals

1: Given domain $\mathcal{D}$ and $\text{LTL}_f/\text{LDL}_f$ formula $\varphi$
2: Compute NFA for $\varphi$ (exponential)
3: Determinize NFA to DFA (exponential)
4: Compute intersection with DFA of $\mathcal{D}$ (polynomial)
5: Compute the belief-state DFA game (exponential)
6: Synthesize winning strategy for resulting DFA game (linear)
7: Return strategy

*Cost is 2-EXPTIME in $\mathcal{D}$ and 3-EXPTIME in $\varphi$!*

Can we do better? **YES!**

## Projection-based Construction

### $\text{POND}_{sp}$ for $\text{LTL}_f/\text{LDL}_f$ goals

### Projection-based Algorithm for $\text{POND}_{sp}$ for $\text{LDL}_f/\text{LTL}_f$ goals

1: Given domain $\mathcal{D}$ and $\text{LTL}_f/\text{LDL}_f$ formula $\varphi$
2: Compute NFA for $\neg\varphi$ (exponential)
4: Compute NFA for union with complement of DFA for $\mathcal{D}$ (polynomial)
3: Project out unobservable props, getting NFA $\overline{A}$ (linear)
*(behaves existentially on unobservables)*
4: Complement NFA $\overline{A}$ getting DFA $A$ (exponential)
*(behaves universally on unobservables)*
5: Synthesize winning strategy for resulting DFA game $A$ (linear)
6: Return strategy

*(Inspired by [DeGiacomoVardi-IJCAI16])*

### Theorem

$\text{POND}_{sp}$ *for* $\text{LTL}_f/\text{LDL}_f$ *goals is:*

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



## Outline

## Conclusion

### Summary

In Planning we separate the **domain** from the **goal** *(this is not the case in synthesis)*, for good reasons!

- **Domain**: it is arepresentation of the world in which the agent acts, hence typically large
  - Cost for $\text{FOND}_{sp}$, $\text{FOND}_{sc}$ is EXPTIME-complete, ...
  - ... independently from the goal being classical reachability, $\text{LTL}_f$ or $\text{LDL}_f$
- **Goal**: it is an objective the agent wants to obtain, hence typically small
  - **Costs depends on the size of the DFA corresponding the $\text{LTL}_f/\text{LDL}_f$ expressing the goal**
  - Polynomial for reachability, i.e., $\Diamond G$, ($G$ propositional), as well as for many $\text{LTL}_f/\text{LDL}_f$ formulas that admit a small (bounded) DFA.
  - Exponential for those $\text{LTL}_f/\text{LDL}_f$ that do not require to determinization
  - 2EXPTIME-complete, in general

### Two basic solvers

We used two basic solvers *on which the planning community has the best know-how*:

- for DFA games ("eventually good"), i.e., a **FOND strong planner**
- for fair DFA games ("eventually good (under fairness)"), i.e., a **FOND strong cyclic planner**

*See work in progress at: http://fond4ltlfpltl.diag.uniroma1.it*

*See papers by Alberto Camacho, Christian Muise, Jorge A. Baier, Sheila A. McIlraith at IJCAI18 and ICAPS18*