Master in Artificial Intelligence and Robotics (AIRO)
Electives in AI
Reasoning Agents

Fabio Patrizi

Sapienza University of Rome, Italy
patrizi@diag.uniroma1.it

A.Y. 2021-2022

Planning for LTL_f and LDL_f Goals*

* slides based on G. De Giacomo's (`www.diag.uniroma1.it/degiacomo`)

Outline

1. $\mathrm{LTL}_f/\mathrm{LDL}_f$: LTL/LDL on finite traces
2. $\mathrm{LTL}_f/\mathrm{LDL}_f$ and automata
3. Planning for $\mathrm{LTL}_f/\mathrm{LDL}_f$ goals: deterministic domains
4. FOND_{sp} for $\mathrm{LTL}_f/\mathrm{LDL}_f$ goals: nondeteministic domains
5. Conclusion

Outline

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) φ holds"
- $\varphi_1 \,\mathcal{U}\, \varphi_2$:"eventually φ_2 holds, and φ_1 holds until φ_2 does"
- $\bullet\varphi \doteq \neg\bigcirc\neg\varphi$: "if next step exists then at next step φ 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 φ 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

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)	$(\neg B \; U A) \lor \Box(\neg B)$
succession(A, B)	*response*$(A, B) \land$ *precedence*(A, B)
alternate response(A, B)	$\Box(A \Rightarrow \bigcirc(\neg A \; U B))$
alternate precedence(A, B)	*precedence*$(A, B) \land$ $\Box(B \Rightarrow \bigcirc($*precedence*$(A, B)))$
alternate succession(A, B)	*alternate response*$(A, B) \land$ *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)	$\neg(\Diamond A \land \Diamond B)$
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$(1, A)$ *existence*$(2, A)$ ... *existence*(n, A)	$\Diamond A$ $\Diamond(A \land \bigcirc($*existence*$(1, A)))$ ... $\Diamond(A \land \bigcirc($*existence*$(n-1, A)))$
absence(A)	$\neg$*existence*$(1, A)$
absence$(2, A)$ *absence*$(3, A)$ ... *absence*$(n+1, A)$	$\neg$*existence*$(2, A)$ $\neg$*existence*$(3, A)$ ... $\neg$*existence*$(n+1, A)$
init(A)	A

LDL$_f$: the language

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

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

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 LTL$_f$:

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

LDL_f: Linear Dynamic Logic on finite traces

LDL_f, not 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 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 (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":

$$\begin{array}{l} [\mathtt{true}^*]\langle \textbf{if } (\mathit{hot}) \textbf{ then} \\ \qquad \textbf{if } (\neg \mathit{airOn}) \textbf{ then } \mathit{turnOnAir} \\ \qquad \textbf{else } \neg \mathit{turnOffAir}\rangle \mathit{true} \end{array}$$

- "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"

$$\begin{array}{l} \langle(\textbf{while } (\mathit{hot}) \textbf{ do} \\ \qquad \textbf{if } (\neg \mathit{airOn}) \textbf{ then } \mathit{turnOnAir} \\ \qquad \textbf{else } \neg \mathit{turnOffAir}; \\ \quad \mathtt{true})^*\rangle \mathit{end} \end{array}$$

Example (LDL_f captures finite domain variant of GOLOG in SitCalc)

GOLOG – finite domain variant

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

- $\pi x.\delta(x)$ **stands for** $\Sigma_{o \in Obj}\ \delta(o)$
- **if** ϕ **then** δ_1**else** δ_2 stands for $(\phi?; \delta_1) + (\neg\phi?; \delta_2)$
- **while** ϕ **do** δ stands for $(\phi?; \delta)^* \neg\phi?$

- $\langle\delta\rangle\phi$ in LDL_f captures SitCalc formula $\exists s'.Do(\delta, s, s') \wedge s \leq s' \leq last \wedge \phi(s')$.
- $[\delta]\phi$ in 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

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 φ is translated into.

We can compile reasoning into automata based procedures!

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 φ (in NNF)

Auxiliary rules

$$
\begin{array}{lll}
\delta(A,\Pi) & = & \texttt{true if } A \in \Pi \\
\delta(A,\Pi) & = & \texttt{false if } A \notin \Pi \\
\delta(\neg A,\Pi) & = & \texttt{false if } A \in \Pi \\
\delta(\neg A,\Pi) & = & \texttt{true if } A \notin \Pi \\
\delta(\varphi_1 \wedge \varphi_2,\Pi) & = & \delta(\varphi_1,\Pi) \wedge \delta(\varphi_2,\Pi) \\
\delta(\varphi_1 \vee \varphi_2,\Pi) & = & \delta(\varphi_1,\Pi) \vee \delta(\varphi_2,\Pi) \\
\delta(\bigcirc\varphi,\Pi) & = & \begin{cases} \varphi & \text{if } Last \notin \Pi \\ \texttt{false} & \text{if } Last \in \Pi \end{cases} \\
\delta(\Diamond\varphi,\Pi) & = & \delta(\varphi,\Pi) \ \vee \ \delta(\bigcirc\Diamond\varphi,\Pi) \\
\delta(\varphi_1 \,\mathcal{U}\, \varphi_2,\Pi) & = & \delta(\varphi_2,\Pi) \ \vee \ (\delta(\varphi_1,\Pi) \wedge \delta(\bigcirc(\varphi_1 \,\mathcal{U}\, \varphi_2),\Pi)) \\
\delta(\bullet\varphi,\Pi) & = & \begin{cases} \varphi & \text{if } Last \notin \Pi \\ \texttt{true} & \text{if } Last \in \Pi \end{cases} \\
\delta(\Box\varphi,\Pi) & = & \delta(\varphi,\Pi) \ \wedge \ \delta(\bullet\Box\varphi,\Pi) \\
\delta(\varphi_1 \,\mathcal{R}\, \varphi_2,\Pi) & = & \delta(\varphi_2,\Pi) \ \wedge \ (\delta(\varphi_1,\Pi) \vee \delta(\bullet(\varphi_1 \,\mathcal{R}\, \varphi_2),\Pi))
\end{array}
$$

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

Algorithm

algorithm LTL$_f$2NFA
input LTL$_f$ formula φ
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 ϱ 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

NFA $\mathcal{A}_\varphi$ associated with an LDL_f formula φ (in NNF)

Auxiliary rules

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

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

Algorithm

algorithm $\mathrm{LDL}_f 2\mathrm{NFA}$
input LDL_f formula φ
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 ϱ 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 (φ SAT)

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

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

1: Given $\text{LTL}_f/\text{LDL}_f$ formula φ
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 Γ and φ
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

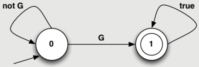
Let Ψ_{domain} describe action domain (LTL_f formula), ϕ_{init} initial state (prop. formula), and G goal (prop. formula). Classical planning amounts to LTL_f satisfiability of:

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

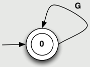
Complexity: PSPACE-complete.

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

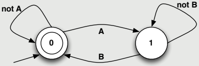
$\Diamond G$



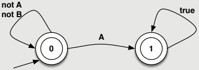
$\Box G$



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

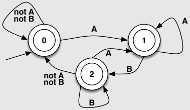


$\neg B\, \mathcal{U}\, A \vee \Box\neg B$ *"A before B"*



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

"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_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

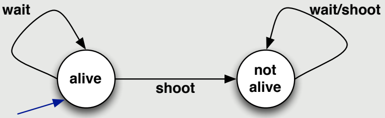
Each trace $s_0, a_1, s_1, \cdots, a_n, s_n$ of the domain $\mathcal{D}$ becomes a finite sequence:

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

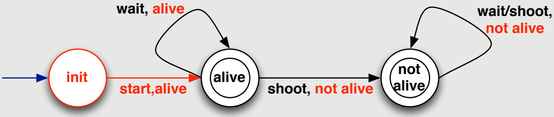
recognized by the DFA $A_{\mathcal{D}}$.

Example (Simplified Yale shooting domain)

- Domain $\mathcal{D}$:



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



Deterministic planning domains as automata

Planning in deterministic domains

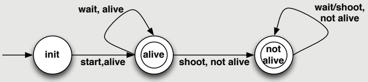
Planning = find a trace of DFA $A_{\mathcal{D}}$ for deterministic domain $\mathcal{D}$ such that is also a trace for the DFA for $\Diamond G$ where G is the goal. That is:

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

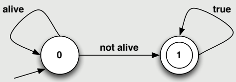
(Computable on-the-fly, PSPACE in $\mathcal{D}$, constant in G. i.e., optimal)

Example (Simplified Yale shooting domain)

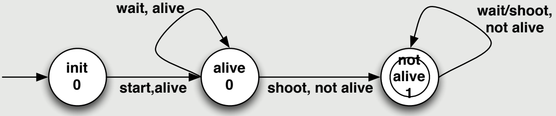
$A_{\mathcal{D}}$



$A_{\Diamond \neg alive}$



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



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

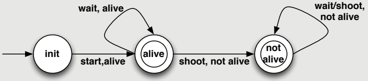
Planning = find a trace of DFA $A_{\mathcal{D}}$ for deterministic domain $\mathcal{D}$ such that is also a accepted by NFA A_φ for the $\text{LTL}_f/\text{LDL}_f$ formula φ. That is:

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

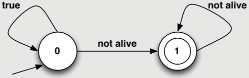
(Computable on-the-fly, PSPACE in $\mathcal{D}$, PSPACE also in φ i.e., optimal)
(We can use NFA directly since we are checking for ***existence*** *of a trace satisfying φ)*

Example (Simplified Yale shooting domain)

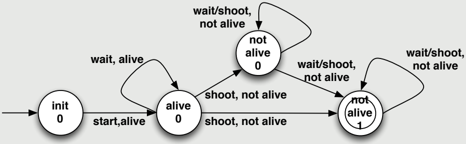
$A_{\mathcal{D}}$



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



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



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 φ
2: Compute corresponding NFA (exponential)
3: Compute intersection with DFA of $\mathcal{D}$ (polynomial)
4: Check nonemptiness of resulting NFA (NLOGSPACE)
5: Return plan

Theorem

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

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

Outline

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.

(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 $A_{\mathcal{D}}$ for $\mathcal{D}$ is a DFA!!!

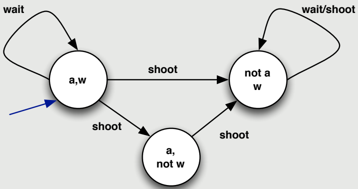
$A_{\mathcal{D}} = (2^{\mathcal{F} \cup \mathcal{A}}, (2^{\mathcal{F}} \cup \{s_{init}\}), s_{init}, \varrho, F)$ where:

- $2^{\mathcal{F} \cup \mathcal{A}}$ 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** $\delta(s, a, s')$ $\qquad \rho(s_{init}, [start, s_0]) = s_0$

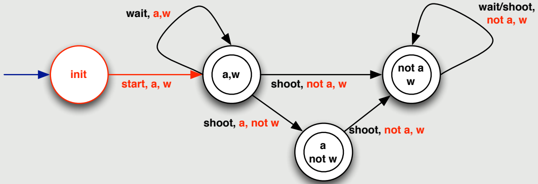
(notation: $[a, s']$ stands for $\{a\} \cup s'$)

Example (Simplified Yale shooting domain variant)

- Domain $\mathcal{D}$:



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

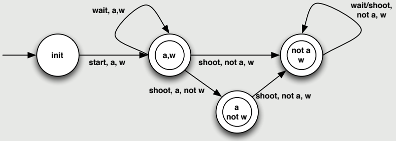


FOND_{sp}: strong planning in nondeterministic domains

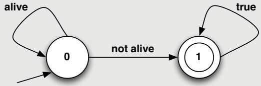
- Set the **arena** formed by all traces that satisfy both the DFA $A_{\mathcal{D}}$ for $\mathcal{D}$ and the DFA for $\Diamond G$ where G is the goal.
- Compute a **winning strategy**. *(EXPTIME-complete in $\mathcal{D}$, constant in G)*

Example (Simplified Yale shooting domain)

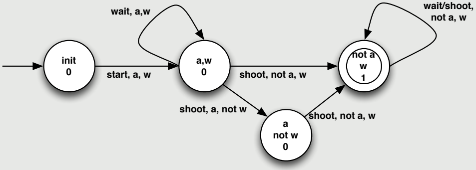
$A_{\mathcal{D}}$



$A_{\Diamond \neg a}$



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

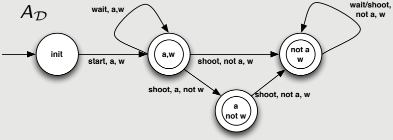


strategy

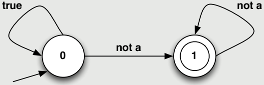
init, 0	$\rightarrow$	*start*
a, w, 0	$\rightarrow$	*shoot*
a, $\neg w$, 0	$\rightarrow$	*shoot*
$\neg a$, w, 1	$\rightarrow$	`win!`

Example (Simplified Yale shooting domain)

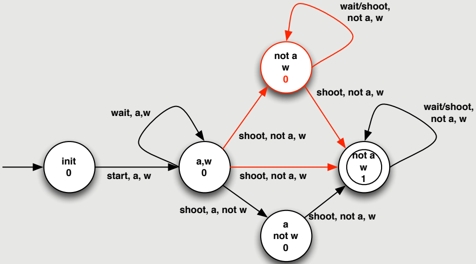
$A_{\mathcal{D}}$



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



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



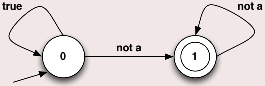
Can we use directly NFA's?

No, because of a basic mismatch

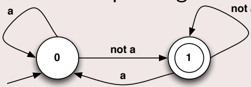
- NFA have perfect **foresight**, or **clairvoyance** *(angelic nondeterminism)*
- Strategies must be runnable: **depend only on past**, not future *(devilish nondeterminism)*

We need first to determinize the NFA for LTL$_f$/LDL$_f$ formula

NFA for $\Diamond\Box\neg a$



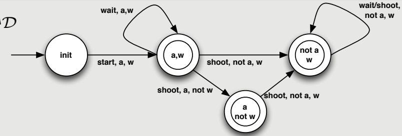
corresponding DFA



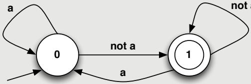
(DFA can be exponential in NFA in general)

Example (Simplified Yale shooting domain)

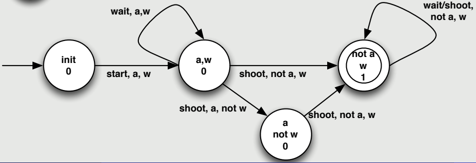
$A_{\mathcal{D}}$



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



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



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;
- $\varrho : S \times 2^{\mathcal{F}\cup\mathcal{A}} \to S$, 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 $f : (2^{\mathcal{F}})^* \to \mathcal{A}$ 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 for all traces π with $a_i = f(\pi_{\mathcal{F}}|_i)$ we have that π 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 ω wrt the domain only, we need to return a **stateful transducer** with transitions from the game.

Generalization: FOND_{sp} for $\text{LTL}_f/\text{LDL}_f$ goals

FOND_{sp} for $\text{LTL}_f/\text{LDL}_f$ goals

Algorithm: FOND_{sp} for $\text{LDL}_f/\text{LTL}_f$ goals

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

Theorem

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

- *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 FOND_{sp}, FOND_{sc} is EXPTIME-complete, ...
  - ... independently from the goal being classical reachability, LTL_f or 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

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