LTL$_f$ Synthesis Under Environment Specification
Planning in Nondeterministic Domains for LTL$_f$ goals

Giuseppe De Giacomo

Outline

Outline

Planning and Synthesis

Planning in Nondeterministic Domain

- Fluents $\mathcal{F}$ (propositions) – controlled by the environment
- Actions $\mathcal{A}$ (actions) – controlled by the agent
- Domain D – specification of the dynamics
- Goal G – propositional formula on fluents describing desired state of affairs to be reached

Planning as game between two players

- Arena: the domain
- Players: agent and environment
- Game: agent tries to force eventually reaching G no matter how other environment reacts
- Problem: find agent-strategy $\sigma_a : (2^{\mathcal{F}})^* \to \mathcal{A}$ to win the game

Complexity

EXPTIME-complete in size of domain specified in PDDL.

Synthesis

- Inputs $\mathcal{X}$ (propositions) – controlled by the environment
- Outputs $\mathcal{Y}$ (propositions) – controlled by the agent
- Domain – not considered
- Goal φ – arbitrary LTL_f (or other temporal logic specification) on both $\mathcal{X}$ and $\mathcal{Y}$

Synthesis as game between two players

- Arena: unconstraint! clique among all possible assignments for $\mathcal{X}$ and $\mathcal{Y}$
- Players: agent and environment
- Game: agent tries to force a play that satisfies φ no matter how other environment reacts.
- Problem: find agent-strategy $\sigma_a (2^{\mathcal{X}})^* \to 2^{\mathcal{Y}}$ to win the game.

Complexity

2EXPTIME-complete in size of φ.

Planning and Synthesis

Planning in Nondeterministic Domain

- Fluents $\mathcal{F}$ (propositions) – controlled by the environment
- Actions $\mathcal{A}$ (actions) – controlled by the agent
- Domain D – specification of the dynamics
- Goal G – propositional formula on fluents describing desired state of affairs to be reached

Planning as game between two players

Arena: the domain

- Players: agent and environment
- Game: agent tries to force eventually reaching G no matter how other environment reacts
- Problem: find agent-strategy $\sigma_a : (2^{\mathcal{F}})^* \to \mathcal{A}$ to win the game

Complexity

EXPTIME-complete in size of domain specified in PDDL.

Synthesis

- Inputs $\mathcal{X}$ (propositions) – controlled by the environment
- Outputs $\mathcal{Y}$ (propositions) – controlled by the agent
- Domain – not considered
- Goal φ – arbitrary LTL_f (or other temporal logic specification) on both $\mathcal{X}$ and $\mathcal{Y}$

Synthesis as game between two players

Arena: unconstraint! clique among all possible assignments for $\mathcal{X}$ and $\mathcal{Y}$

- Players: agent and environment
- Game: agent tries to force a play that satisfies φ no matter how other environment reacts.
- Problem: find agent-strategy $\sigma_a (2^{\mathcal{X}})^* \to 2^{\mathcal{Y}}$ to win the game.

Complexity

2EXPTIME-complete in size of φ.

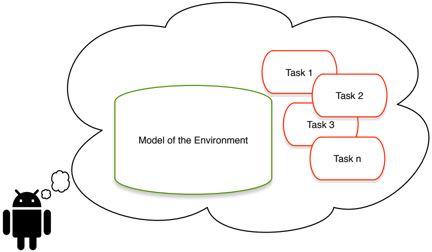
We want to revisit the assumption that the environment is unconstrained!

Outline

Planning in Nondeterministic Domains (FOND_{sp})

Planning in nondeterministic domains

- Environment Model (DOM)
  - Environment model is called "domain"
  - Specs of environment's behaviors of the world in response to agent's action
  - Domain expressed as with specific formalisms
    - PDDL
  - DOM is, or better generates, a non-deterministic transition system, i.e., a **game arena for two players Agent and Env**!
- Agent Task (GOAL)
  - Agent task is called "goal"
  - Specs of task to achieve
  - GOAL expressed as reaching a state of the domain with desired properties
- Find agent plan/program/strategy/policy that fulfills GOAL in DOM



Find plan that fulfills the desired task
in spite of how the environment responds,
i.e., wins the GOAL in nondeterministic DOM

Planning in Nondeterministic Domains

Transition system induced by a nondeterministic domain

A nondeterministic domain $D = (\mathcal{F}, \mathcal{A}, I)$ induces a transition system $T_D = (2^{\mathcal{F}}, \mathcal{A}, s_0, \alpha, \delta)$ where:

- $\mathcal{F}$ is the set of fluents (atomic propositions)
- $\mathcal{A}$ is the set of actions (atomic symbols)
- $2^{\mathcal{F}}$ is the set of states
- s_0 is the initial state (initial assignment to fluents)
- $\alpha(s) \subseteq \mathcal{A}$ represents action preconditions
- $\delta(s, a, s')$ with $a \in \alpha(s)$ represents nondeterministic action effects (including frame).
  Note the transition function is now a transition relation, i.e., given a state s and an action a we have a set of possible successor states $\{s' \mid \delta(s, a, s')\}$.

Planning in Nondeterministic Domains

Who controls what?

Fluents controlled by **environment**

Actions controlled by **agent**

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

Game arena induced by a nondeterministic domain

If we consider this information on the control, then T_D is in fact a **game arena**: $T_D = (2^{\mathcal{F}}, \mathcal{A}, s_0, \alpha, \delta)$ where:

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

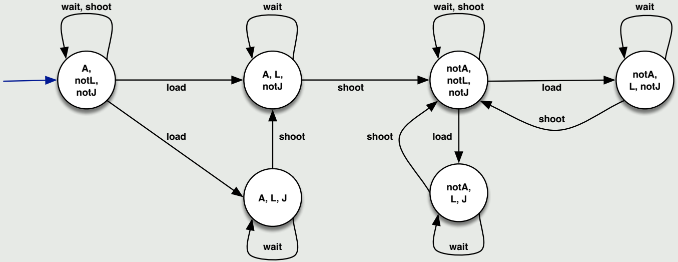
Hence to execute a transition in state s

- The agent needs to choose the action a
- The environment need to choose the resulting state s'.

Nondeterministic Yale Shooting Example

Yale shooting game arena

Yale Shooting domain D induces the following game arena T_D:



Planning in Nondeterministic Domain

Planning

Given a nondeterministic domain D and a goal G in propositional logic:

- Find agent executable strategy (or plan) σ_a such that for every environment strategies σ_e that are compliant with D, we have that the $play(\sigma_a, \sigma_e) = s_0, a_1, s_1, \ldots, s_{n-1}, a_n, s_n$ is such that $s_n \models G$.

In other words find a strategy (or plan) σ_a that reaches a state where G holds no matter what the environment does.

The strategy σ_a, if it exists is called **winning strategy**

LTL_f Goals

(1) In order express arbitrary goals in LTL_f –or LTL–, we need to:

Represent actions as propositions

Decide how we represent actions as propositions of LTL_f formulas.

- Use one proposition a for each action a. Then:
  - We need to add the requirement that at most one action proposition is true in each instant $\Box(a \supset \bigwedge_{b \in A \wedge b \neq a} \neg b)$.
- Use a binary (logarithmic) encoding of action each a. Then:
  - Each action a is represented as a boolean formula a over the propositions for the binary encoding;
  - Some binary encoding will correspond to non-existing actions, if the number of actions is not a power of 2. In this case we need to specify what these spurious action do in the transition system, e.g., $nope$, or we need to forbid them.

For now, we will adopt the first way of representing actions, but later when we study symbolic technique we will also use the latter.

(cf. LTL_f Model Checking)

LTL$_f$ Goals

(2) In order express arbitrary goals in LTL$_f$ –or LTL–, we also need to:

Pair actions and states in a time instant

Decide how we need to pair actions and states in a time instant

- Pair the agent action and the resulting state, (in fact labeling of the state) of the environment

  The propositional representation a for an action a will stand for "action a just executed".

- Pair current environment (labeling of the) state and the next action instructed by the agent

  The propositional representation a for an action a will stand for "action a just instructed to be executed next".

Both ways of pairing actions and states are fine. But choosing one or the other is essential, because it changes how we specify properties in LTL$_f$.

(cf. LTL$_f$ Model Checking)

LTL$_f$ Goals

LTL$_f$-traces

A T_D trace $s_0, a_1, s_1, \cdots, a_n, s_n$ induces a corresponding LTL$_f$-trace:

- If we pair action and the resulting state: $(dummy, s_0), (a_1, s_1), \cdots, (a_n, s_n)$, where $dummy$ is a dummy starting action.
- If we pair state and the next action: $(s_0, a_1), (s_1, a_2) \cdots, (s_{n-1} a_n), (s_n, dummy)$, where $dummy$ is a dummy ending action.

Example

The way we pair actions and states changes how we specify properties in LTL$_f$:
Suppose we want to say:

every time that ϕ_1 is true in the current state if we do action a we get ϕ_2 in the next state".

- If we pair action and the resulting state, we write: $\Box(\phi_1 \supset \bullet(a \supset \phi_2))$
- If we pair state and the next action, we write: $\Box((\phi_1 \wedge a) \supset \bullet\phi_2)$

In this course we pair action and the resulting state to have traces that represents cleanly histories (things already happened).

Nondeterministic Domains as Automata

With these decisions taken we can transform the nondeterministic domain $T_D = (2^{\mathcal{F}}, \mathcal{A}, s_0, \alpha, \delta)$ into an automaton recognizing all its traces.

Automaton A_D for D is a DFA!!!

$A_D = (2^{\mathcal{F} \cup \mathcal{A}}, Q, q_{init}, \rho, F)$ where:

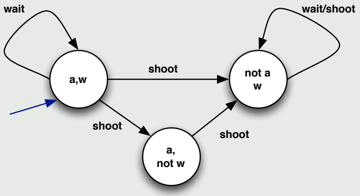
- $2^{\mathcal{F} \cup \mathcal{A}}$ alphabet (actions $\mathcal{A}$ include dummy $start$ action)
- $Q = 2^{\mathcal{F}} \cup \{q_{init}\}$ set of states
- q_{init} dummy initial state
- $F = 2^{\mathcal{F}}$ (all states of the domain are final)
- $\rho(s, [a, s']) = s'$ **with** $a \in \alpha(s)$**,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

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

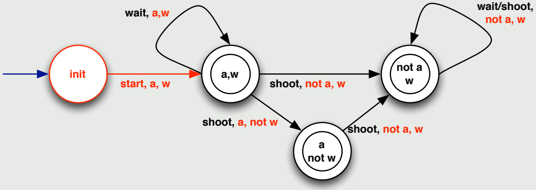
Nondeterministic Domains as Automata

Example (Simplified Yale shooting domain variant)

- Domain $\mathcal{T}_D$:



- DFA A_D:



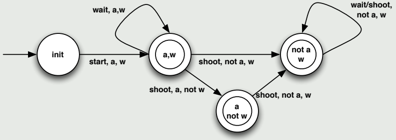
Nondeterministic Domains as Automata

Planning in nondeterministic domains

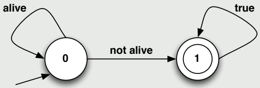
- Set the **arena** formed by all traces that satisfy both the DFA A_D for D and the DFA for $\Diamond G$ where G is the goal.
- Compute a **winning strategy**. *(EXPTIME-complete in D, constant in G)*

Example (Simplified Yale shooting domain)

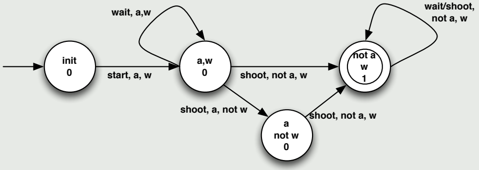
A_D



$A_{\Diamond\neg a}$



$A_D \cap A_{\Diamond\neg a}$:



strategy

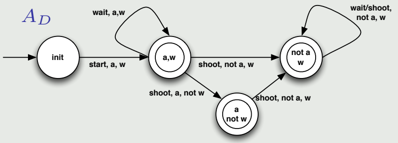
$init, 0$	$\rightarrow$	$start$
$a, w, 0$	$\rightarrow$	$shoot$
$a, \neg w, 0$	$\rightarrow$	$shoot$
$\neg a, w, 1$	$\rightarrow$	win!

FOND_{sp} for LTL_f Goals

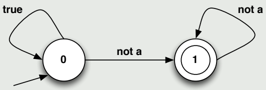
Example (Simplified Yale shooting domain)

Consider the goal $\Diamond\Box\neg a$.

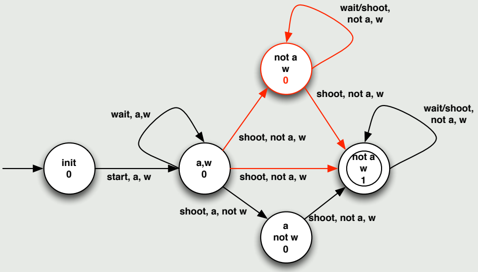
A_D



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



$A_D \cap A_{\Diamond\Box\neg a}$:



Can we use directly NFA's?

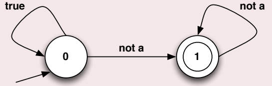
No, because of a basic mismatch

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

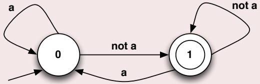
FOND_{sp} for LTL_f Goals

We need first to determinize the NFA for LTL_f formula

NFA for $\Diamond\Box\neg a$



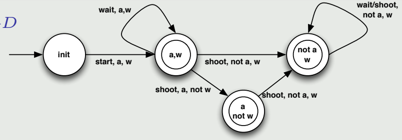
corresponding DFA



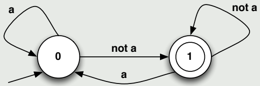
(DFA can be exponential in NFA in general)

Example (Simplified Yale shooting domain)

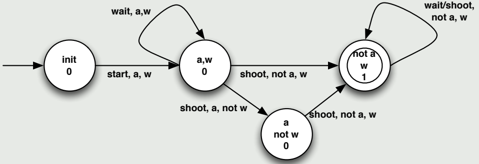
A_D



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



$A_D \cap A_{\Diamond\Box\neg a}$:



strategy

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

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 $\mathcal{G}$.

DFA Games

Winning condition for DFA games

Let

$$PreAdv(\mathcal{S}) = \{s \in S \mid \exists a \in \alpha(s).\, \forall e \in 2^{\mathcal{F}}.\, \varrho(s, [a, e]) \in \mathcal{S}\}$$

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 $\mathcal{G}$)
- $Win_{i+1} = Win_i \cup PreAdv(Win_i)$
- $Win = \bigcup_i Win_i$

(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.

FOND_{sp} for LTL_f Goals

FOND_{sp} for LTL_f goals

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

1: Given LTL_f domain D and goal φ
2: Compute NFA for φ (exponential)
3: Determinize NFA to DFA (exponential)
4: Compute intersection with DFA of D (polynomial)
5: Synthesize winning strategy for DFA game (linear)
6: Return strategy

Theorem

Planning in nondetermnistic domains for LTL_f goals is:

- *EXPTIME-complete in the domain (compactly represented using of fluents – polynomial in number of states);*
- *2-EXPTIME-complete in the goal.*