Game-Theoretic Approach to Planning and Synthesis

Planning and Synthesis

Giuseppe De Giacomo Antonio Di Stasio Giuseppe Perelli Shufang Zhu

ERC Advanced Grant
WhiteMech:
White-box Self Programming Mechanisms

PhD-AI Course
4-8 July, 2022

Autonomy is one of the grand objectives of AI.

Autonomy in AI

Aims at building autonomous agents/robots that operate in changing, incompletely known, unpredictable environments.

- Aims at empowering the agent with the ability of ...
- deliberating ...
- how to act in its environment ...
- autonomously (without human intervention)

First person view

It is the agent/system that decides what to do!

Concrete Examples of Autonomous Systems

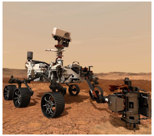
Space Exploration

Delay in communication requires high-level of autonomy during the mission.

Planning and scheduling for temporal extended goals is a top research topic at NASA.

https://mars.nasa.gov/mars2020/

NASA Science MARS 2020 MISSION PERSEVERANCE ROVER



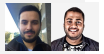
Interacting with Humans

Robots interacting with humans needs to properly schedule execution of complex actions that extends over time, as well as react to unexpected circumstances.

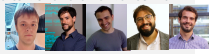
RoboCup

Autonomy is a Challenge in Business Process Management (BPM)

- BPM'21 Tutorial: **Applications of Automated Planning for Business Process Management** by *Andrea Marrella and Tathagata Chakraborti*



- BPM'21 Tutorial: **RuM: Declarative Process Mining, Distilled** by *Anti Alman, Claudio Di Ciccio, Fabrizio Maria Maggi, Marco Montali and Han van der Aa*



- BPM'21 Invited talk: **Artificial Intelligence-based Declarative Process Synthesis for BPM** by *Giuseppe De Giacomo*

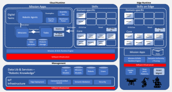
M. Dumas. Constructing Digital Twins for Accurate and Reliable What-If Business Process Analysis. PROBLEMS'21@BPM'21

Autonomous Mobile Robots in Logistics

Complex multi-robot systems need highly synchronized behaviours to fulfil their job.

These robots need autonomously resolve unexpected clashes.

Sophisticated AMR platforms under study by industry



FinTech

- Smart Contracts Composition: systematic approach to compose many Smart Contracts (e.g. on Ethereum) from an extracted abstract specification to implement novel services for the user.
- Encoding Trading Strategies with Formal Methods:
  - **Safety conditions:** stop-loss, take-profit
  - **Liveness conditions:** order placement control
  - Interoperability with learning modules can forecast optimal order amounts
- Security & Fraud Detection: Process Mining to discover suspicious patterns of illegal activities e.g. money laundering
- Many AI + FinTech Academic Venues!
  - ICAPS's FinPlan workshop
  - IJCAI's FinTech track

ethereum

CHASE J.P.Morgan *2021AI Faculty Award*

Smart Manufacturing and Digital Twins

- Manufacturing as a service products to be manufactured are not known in advance and each product may differ from the products manufactured immediately before and immediately after it
- Analogies with Service Composition and Orchestration: synthesize the orchestrator
- Digital Twins platforms offer infrastructure to deploy orchestrations
- Automated exception handling is crucial

G. De Giacomo, P. Felli, B. Logan, F. Patrizi, S. Sardina: Situation Calculus for Controller Synthesis in Manufacturing Systems with First-Order State Representation. AIJ2021

G. De Giacomo, M. Favorito, F. Leotta, M. Mecella, L. Silo: Digital Twins Composition via Markov Decision Processes. ITBPM2021

- Autonomy requires:
  - Reasoning and planning capabilities (this course)
  - Learning from experience (see later course in Non-Markovian RL)
- Many areas of AI are concern with autonomy:
  - Logics in AI
  - Knowledge representation and reasoning
  - Planning
  - Multi-agent systems
  - Sequential decision making (MDPs)
  - Reinforcement learning
- Recently: Some objectives are shared with automated program synthesis in Formal Methods

WhiteMech: Whitebox Self Programming Mechanisms
ERC Advanced Grant

Formal Methods

- Aims at giving rigorous guarantees on the behavior of processes/dynamic systems
- Wide-spread industrial adoption
- Based on formal logic
- Developed formalisms/logics for expressing complex properties of processes/dynamic systems
- Main tasks
  - Verification: given a system and a specification, check if the system satisfies the specification.
  - Automated program synthesis: given a specification, synthesize a system that satisfies it.

Basic Idea: "Mechanical translation of human-understandable task specifications to a program that is known to meet the specifications." *[Vardi - The Siren Song of Temporal Synthesis 2018]*

- Classical Synthesis: Synthesize transformational programs *[Green1969], [WaldingerLee1969], [Manna and Waldinger1980]*
- Reactive Synthesis: Synthesize programs for interactive/reactive ongoing computations (protocols, operating systems, controllers, robots, etc.) *[Church1963], [HarelPnueli1985], [AbadiLamportWolper1989], [PnueliRosner1989], [EhlersLafortuneTripakisVardi2017], [Finkbeiner2018]*

We focus on Reactive Synthesis (which we call simply synthesis form now on)

Synthesis

:

- Agent acts in a (nondeterministic) Environment
- Agent controls **actions**
- Environment controls **fluents**
- Task is given to agent
- Task talks both about fluents and actions
- Agent has to realize the task in spite of how the Environment reacts.

Similar to planning in particular similar to planning in fully observable nondeterministic domains (FOND)
[DeGiacomoVardi2015], [DeGiacomoVardi2016], [DeGiacomoRubin2018], [CamachoTriantafillouMuiseBaierMcIlraith2017], [CamachoMuiseBaierMcIlraith2018], [CamachoBienvenuMcIlraith2019]

Agent process/behavior

Agent process/behavior *(also called, "plan", "strategy", "policy", "protocol")*:

$$\sigma_a : (\mathit{fluents})^* \rightarrow \mathit{actions}$$

where

- $(\mathit{fluents})^*$ denotes the history of what observed so far by the agent

  (a finite sequence of fluents configurations)

- *actions* denotes the next action that the agent does

Every program/process has this form! [AbadiLamportWolper89].

Environment process/behavior

Environment process/behavior:

$$\sigma_e : (actions)^* \rightarrow fluents$$

where

- $(actions)^*$ denotes the history of what observed so far by the environment

  (a finite sequence of agent actions)

- *fluents* denotes the next effects that the environment brings about.

Every program/process has this form! [AbadiLamportWolper89].

Traces

Observe that both the agent process and the environment process:

$$\sigma_a : \quad (fluents)^* \quad \rightarrow \quad actions$$

$$\sigma_e : \quad (actions)^* \quad \rightarrow \quad fluents$$

cannot be executed in isolation.

But they can be executed together, generating a trace *(sometime also call a "play")*:

$$trace(\sigma_a, \sigma_e) = F_0 \cup A_0; F_1 \cup A_1; \cdots$$

In Formal Methods one of the most used formalism to specify properties of processes is Linear Time Logic (LTL) *[Pnueli 1977]*.

LTL

LTL Syntax

$$\varphi ::= A \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \mathit{next}\,\varphi \mid \mathit{eventually}\,\varphi \mid \mathit{always}\,\varphi \mid \varphi_1\ \mathit{until}\ \varphi_2$$

LTL Semantics

A **trace** τ is an infinite (LTL) sequence fluents $\cup$ actions evaluations. Write $\tau \vDash \varphi$ to mean that φ is true in τ.

Example (LTL for properties of processes)

$$\mathit{eventually}\,\mathit{InRoom101}$$

$$\mathit{always}\,\mathit{InSafeArea}$$

$$\mathit{eventually}\,\mathit{InRoom101} \wedge \mathit{always}\,\neg\mathit{PreViolated}$$

$$\mathit{always}(\mathit{CoffeeRequested} \supset \mathit{eventually}\,\mathit{CoffeeDelivered})$$

$$\mathit{always}(\mathit{openDoor} \supset \mathit{next}\,\mathit{closeDoor})$$

Notice that we are actually constraining the interaction of the agent with the environment!

How can we formalize (reactive) synthesis?

Synthesis

Given an LTL task *Task* for the agent, find agent process, i.e., strategy σ_a such that for whatever possible environment process σ_e we have $trace(\sigma_a, \sigma_e) \models Task$, i.e.,

$$\text{find } \sigma_a \text{ such that } \forall \sigma_e . trace(\sigma_a, \sigma_e) \models Task$$

Note: How do we restrict the possible processes/behavior of the Environment?
This has been of primary interest in AI!

Knowledge Representation

"Equip agent with a model of its environment"

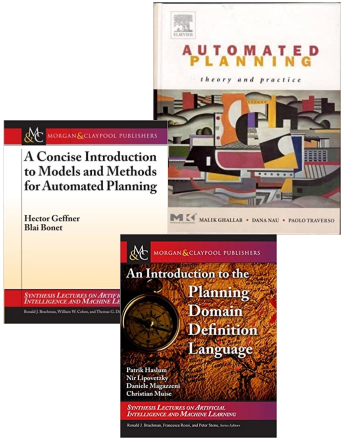
Reasoning about Actions (classical view):

- Capture model of the environment with a logical theory
  - Preconditions for agent actions
  - Effects of agent actions + solution to "Frame Problem"
- Multiple interpretations of the theory
  - Multiple possible instantiations of the environment (one of them the correct one, but we do not know which)
- Reasoning (skeptical)
  - based on logical implication

Planning

Share the same view as KR:
"Equip agent with a model of its environment"

- Inherit from KR the idea that environment can be represented in terms of
  - Preconditions for agent actions
  - Effects of agent actions + solution to "Frame Problem"
- But use them as a specification for generating a transition system
  - a single model vs a theory
- Reasoning (skeptical)
  - based on "model checking"
- Two notable cases:
  - Classical planning: everything determined by agent actions
  - Planning in nondeterministic domains (FOND):
    - Agent: instructs actions
    - Env: determine their effects



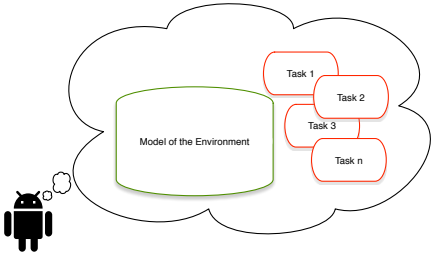
SAPIENZA UNIVERSITÀ DI ROMA

Tasks in planning

- Planning in AI is all about
  - specifying a "goal", i.e., a task ...
  - and producing a "plan", i.e., a program, strategy, or policy,
  - that satisfies the task in the environment model.
- Which tasks?
  - **Tasks that terminates!**
  - Typically, just reaching a certain state in the environment

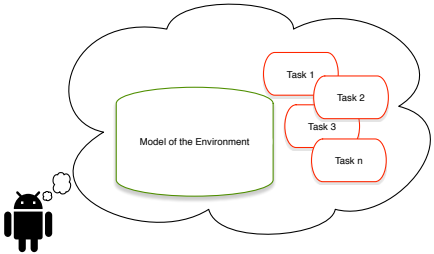
Why tasks that terminates?

- Because it is the agent that is planning/reasoning
- If the task would not terminate, the agent would be stuck into doing the same task forever
- But then, why bother with equipping it with autonomous reasoning abilities at all?
- We want to focus on autonomous intelligent agents that (1) get a task, (2) reason/plan autonomously to solve it, (3) execute the plan, (4) get another task, and so on.



Planning in deterministic 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
    - STRIPS
    - ADL
    - PDDL
  - DOM is, or better generates, a deterministic transition system
- 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 the model of the environment, i.e., wins GOAL in DOM

Deterministic domain (including initial state)

$\mathcal{D} = (\mathcal{S}, \mathcal{A}, s_0, \delta, \alpha)$ where: *It's a transition system!*

- $\mathcal{F}$ **fluents** (atomic propositions)
- $\mathcal{A}$ **actions** (atomic symbols)
- $\mathcal{S} = 2^{\mathcal{F}}$ 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 $a_1, \cdots, a_n$.

Example (Yale shooting scenario - deterministic variant)

Consider the following simplified variant of the Yale Shooting Scenario, which has to do with a shooting a turkey named Fred. Find a plan to kill the turkey.

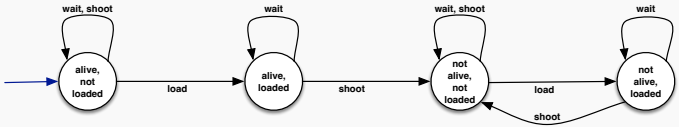
- Fluents:
  - *alive* - The turkey is alive in the current situation;
  - *loaded*– The gun is loaded in the current situation.
- Actions:
  - *load* – Load the gun.
    - PRE: Requires that the gun is not loaded;
    - EFF: The gun is loaded
  - *shoot* - Shoot the gun.
    - PRE: Can always be performed;
    - EFF: If the gun is loaded then it unloads the gun and kills the turkey (not alive), otherwise has no effects
  - *wait* – A no-op;
    - PRE: Can always be performed;
    - EFF: has no effect on any fluent

Note: every fluent not mentioned in the effect of actions remains unchanged (frame problem)

- Initial situation description:
  - Initially the turkey is alive, and the gun is not loaded.
- Goal:
  - Kill the turkey, i.e., reach a situation where the turkey is not alive.

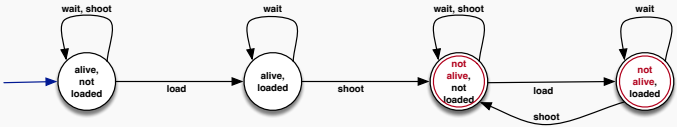
Example (Yale shooting domain)

Yale Shooting domain $\mathcal{D}$ as a transition system:



Example (Yale shooting domain with goal)

Consider the goal: $\neg alive$. The domain $\mathcal{D}$ with the goal states highlighted is the following:



Winning states for deterministic domains

Let denote the **set of goal states** as:

$$[\![G]\!] = \{s \in \mathcal{S} \mid s \vDash G\}$$

and let's define the **(existential) preimage of a set** $\mathcal{E}$ the following function:

$$PreE(\mathcal{E}) = \{s \in \mathcal{S} \mid \exists a \in \alpha(s).\delta(s,a) \in \mathcal{E}\}$$

Compute the set Win of winning states of the domain for goal G, i.e., states from which the agent can reach the goal G, by **least-fixpoint**:

- $Win_0 = [\![G]\!]$ (the goal states)
- $Win_{i+1} = Win_i \cup PreE(Win_i)$
- $Win = \bigcup_i Win_i$

$W_{old} := \emptyset$
$W := [\![G]\!]$
while $(W \neq W_{old})\{$
  $W_{old} := W$
  $W := W \cup PreE(W)$
$\}$
return W

(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:

$$\omega(s) = \{a \in \alpha(s) \mid \text{ if } s \in Win_{i+1} - Win_i \text{ then } \delta(s,a) \in Win_i\}$$

Every way of restricting $\omega(s)$ to return only one action (chosen arbitrarily) gives a **winning strategy** to reach G, i.e., a **plan**.

- The kind of plan just computed is called **"Universal Plan"**
  - It is able to say what to do to win in every single state!
  - If for any reason during the execution of the plan the agent find itself in an unexpected state
    - i.e.: the agent expected an effect but s/he got another one

  Then the universal plan knows what to do to win anyway

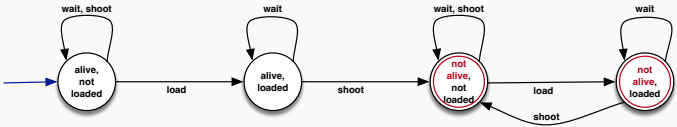
- Typically in planning it is sufficient to have a classical plan (not a universal one), i.e., a sequence of actions that from the initial state leads the agent to a state satisfying the goal.

- To compute classical plans there are better algorithms (in fact exceptionally fast) – based on heuristic forward search (see any course in Planning)

- In this course we will focus on **"Universal Plans"**, aka programs/strategies/policies

Example (Yale shooting domain with goal)

Consider the goal: $\neg alive$. The domain $\mathcal{D}$ with the goal states highlighted is the following:



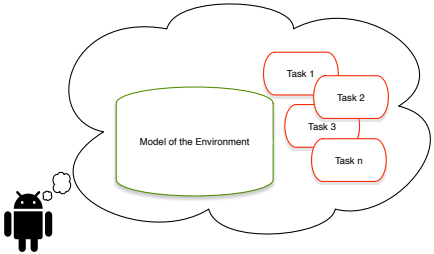
Exercise: Compute a plan for achieving the goal.

SAPIENZA UNIVERSITÀ DI ROMA

SAPIENZA UNIVERSITÀ DI ROMA

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

SAPIENZA UNIVERSITÀ DI ROMA

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)*

Example (Yale shooting scenario – nondeterministic variant)

Consider the following nondeterministic variant of the Yale Shooting Scenario.

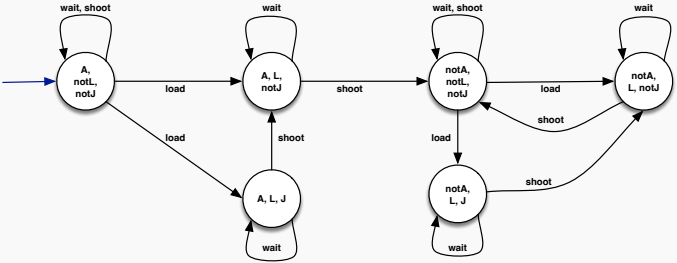
- Fluents:
  - *alive* - The turkey is alive in the current situation;
  - *loaded*– The gun is loaded in the current situation;
  - *jammed*– The gun is jammed in the current situation.
- Actions:
  - *load* – Load the gun.
    - PRE: Requires that the gun is not loaded;
    - EFF: It has two alternative possible effects:
      Either (*i*) it loads the gun and does not jam it; or (*ii*) it loads the gun and jams it.

    (Use PDDL "oneof" for expressing alternative effects.)
  - *shoot* - Shoot the gun.
    - PRE: Can always be performed;
    - EFF: If the gun is loaded and not jammed then it unloads the gun and kills the turkey (not alive), if the gun is loaded and jammed then it unjams the gun; if the gun is not loaded then it does nothing.
  - *wait* – A no-op;
    - PRE: Can always be performed;
    - EFF: has no effect on any fluent

  Note: every fluent not mentioned in the effect of actions remains unchanged (frame problem)
- Initial situation description:
  - Initially the turkey is alive, the gun is not loaded, and the gun is not jammed.
- Goal:
  - Kill the turkey, i.e., reach a situation where the turkey is not alive.

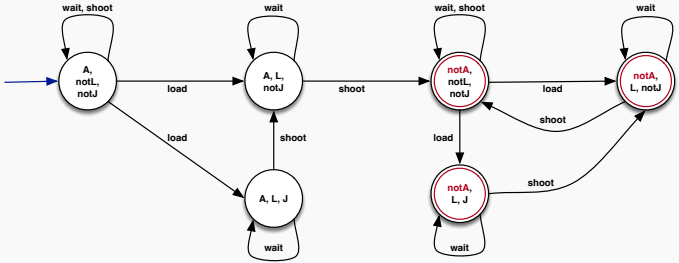
Example (Yale shooting domain)

Yale Shooting domain $\mathcal{D}$ as a transition system:



Example (Yale shooting domain with goal)

Consider the goal: $\neg alive$. The domain $\mathcal{D}$ with the goal states highlighted is the following:



Winning states for nondeterministic domains

Let denote the **set of goal states** as:

$$[\![G]\!] = \{s \in \mathcal{S} \mid s \vDash G\}$$

and let's define the **(adversarial preimage of a set $\mathcal{E}$** the following function:

$$PreAdv(\mathcal{E}) = \{s \in \mathcal{S} \mid \exists a \in \alpha(s).\, \forall s' \in \mathcal{S}.\, \delta(s, a, s') \supset s' \in \mathcal{E}\}$$

Compute the set *Win* of winning states of the domain for goal G, i.e., states from which the agent can reach the goal G, by **least-fixpoint**:

- $Win_0 = [\![G]\!]$ (the goal states)
- $Win_{i+1} = Win_i \cup PreAdv(Win_i)$
- $Win = \bigcup_i Win_i$

$W_{old} := \emptyset$
$W := [\![G]\!]$
while $(W \neq W_{old})\{$
  $W_{old} := W$
  $W := W \cup PreAdv(W)$
$\}$
return W

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

Computing the winning strategy

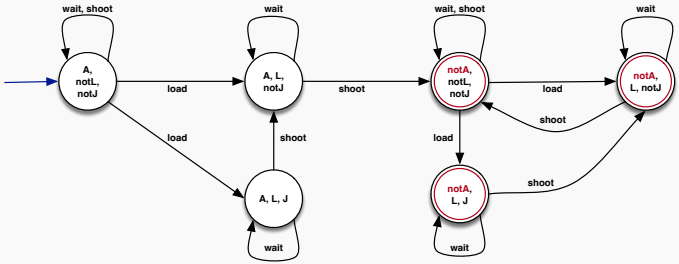
Let's define $\omega : S \to 2^{\mathcal{A}}$ as:

$$\omega(s) = \{a \in \alpha(s) \mid \text{ if } s \in Win_{i+1} - Win_i \text{ then } \forall s'.\delta(s, a, s') \supset s' \in Win_i\}$$

Every way of restricting $\omega(s)$ to return only one action (chosen arbitrarily) gives a **winning strategy** to reach G, i.e., a **plan**.

Example (Yale shooting domain with goal)

Consider the goal: $\neg alive$. The domain $\mathcal{D}$ with the goal states highlighted is the following:

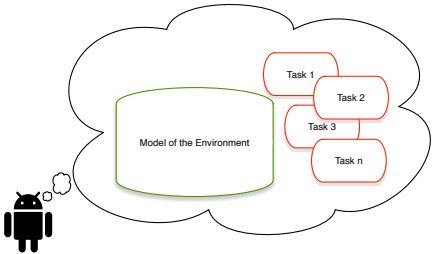


Exercise: Compute a plan for achieving the goal in spite of the environment nondeterminism.

SAPIENZA UNIVERSITÀ DI ROMA

Planning in nondeterministic stochastic fair 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!
  - **Env acts stochastically and not adversarially**,
  - i.e., Env is **"stochastic fair"**
- 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, knowing that Env acts stochastically, i.e., wins the GOAL in nondeterministic, stochastic fair *DOM*

SAPIENZA UNIVERSITÀ DI ROMA

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.

(FOND_{SC} *is EXPTIME-complete)*

SAPIENZA UNIVERSITÀ DI ROMA

Example (Yale shooting scenario – stochastic fair variant)

Consider the following nondeterministic, stochastically fair variant of the Yale Shooting Scenario.

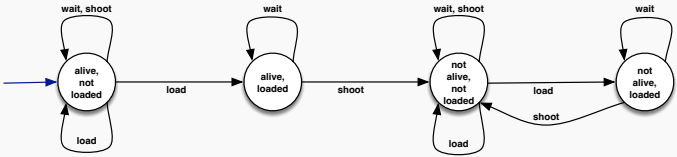
- Fluents:
  - *alive* - The turkey is alive in the current situation;
  - *loaded*– The gun is loaded in the current situation;
- Actions:
  - *load* – Load the gun.
    - PRE: Requires that the gun is not loaded;
    - EFF: It has two alternative possible effects:
      Either (i) it loads the gun; or (ii) does nothing.

    (Use PDDL "oneof" for expressing alternative effects.)
  - *shoot* - Shoot the gun.
    - PRE: Can always be performed;
    - EFF: If the gun is loaded then it unloads the gun and kills the turkey (not alive), otherwise does nothing.
  - *wait* – A no-op;
    - PRE: Can always be performed;
    - EFF: has no effect on any fluent

  Note: every fluent not mentioned in the effect of actions remains unchanged (frame problem)
- Initial situation description:
  - Initially the turkey is alive, and the gun is not loaded
- Goal:
  - Kill the turkey, i.e., reach a situation where the turkey is not alive.

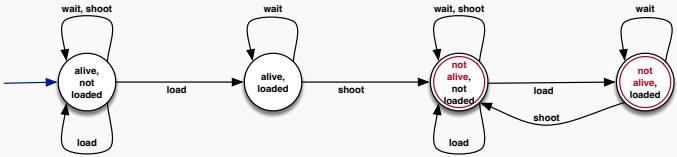
Example (Yale shooting domain)

Yale Shooting domain $\mathcal{D}$ as a transition system:



Example (Yale shooting domain with goal)

Consider the goal: $\neg alive$. The domain $\mathcal{D}$ with the goal states highlighted is the following:



Existential and universal preimages wrt the environment

$$PreE(a, \mathcal{E}) = \{s \in S \mid a \in \alpha(s).\ \exists s' \in S.\ \varrho(s, a, s') \wedge s \in \mathcal{E}\}$$
$$PreA(a, \mathcal{E}) = \{s \in S \mid a \in \alpha(s).\ \forall s' \in S.\ \varrho(s, a, s') \supset s' \in \mathcal{E}\}$$

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):

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

This gives rise to the following nested fixpoint computation:

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

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

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

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

Computing the winning strategy

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

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

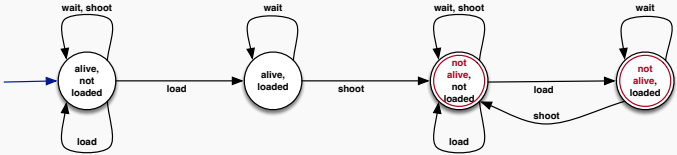
Note that $Safe = \cup_{j \leq |S|} Reach_j$.
Let's define $\omega : S \to 2^{\mathcal{A}}$ as:

$$\omega(s) = \{a \in \alpha(s) \mid \text{ if } s \in Reach_{j+1} - Reach_j \text{ then } \exists s' \in \mathcal{S}.\varrho(s, a, s') \wedge s' \in Reach_j\}$$

Every way of restricting $\omega(s)$ to return only one action (chosen arbitrarily) gives a **winning strategy**, i.e., a **plan**, to eventually reach G (with the help of the stochasticity of the environment).

Example (Yale shooting domain with goal)

Consider the goal: $\neg alive$. The domain $\mathcal{D}$ with the goal states highlighted is the following:



Exercise: Compute a plan for achieving the goal in spite of the environment nondeterminism, considering that the environment is stochastic fair.