From PAs to POMDPs: Making Optimal Decisions Under Uncertainty

Pierre Vandenhove

UMONS – Université de Mons

Theoretical CS Seminar @ UA – March 11, 2026

Outline

Outline

The need for stochasticity in computer science

You have probably seen multiple **computational models** in your classes:

- **automata**,
- Boolean circuits,
- Turing machines,
- ...

Why include **stochasticity** in our models? **Two reasons** (at least):

- **Efficiency**: Some computations can be done more efficiently with randomness. E.g.,
  - taking random pivots in quicksort,
  - testing whether a number is prime using Miller-Rabin.
- **Modelling power**: To model systems that naturally exhibit randomness. E.g.,
  - modelling epidemics,
  - modelling financial markets,
  - modelling noisy communication channels.

Common models

- Let us start from the model of **graphs**.
- To make them **more expressive** (and interesting), two common ingredients are added:
  - **stochasticity** and
  - **control**: some agent makes decisions and tries to achieve some goal.

	No control	**Control**
No stochasticity	Graphs	(Non)det. automata
Stochasticity	Markov chains	**Probabilistic automata**

Outline

Probabilistic automata: definition

A **probabilistic automaton**[1] (PA) is a tuple $\mathcal{A} = (Q, \Sigma, q_{\text{init}}, \delta, F)$ such that:

- Q is a finite set of states,
- Σ is a finite alphabet,
- $q_{\text{init}} \in Q$ is the initial state,
- $\delta \colon Q \times \Sigma \to \text{Dist}(Q)$ is the transition function,
- $F \subseteq Q$ is the set of accepting states.

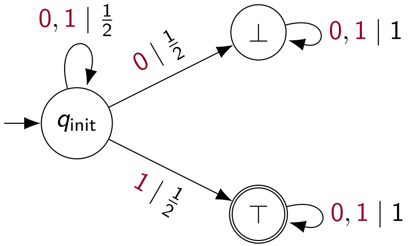
It **generalizes** other models:

- A **deterministic automaton** corresponds to the case where $\delta : Q \times \Sigma \to Q$.
- A **nondeterministic automaton** corresponds to $\delta : Q \times \Sigma \to 2^Q$.
- A **Markov chain** corresponds to $|\Sigma| = 1$.

[1] Rabin, "Probabilistic Automata", 1963.

Probabilistic automaton: example

Let $\Sigma = \{0, 1\}$. Consider the following[2] PA $\mathcal{A}_{\text{bin}}$:



For the transition function δ, we can equivalently give the transition matrices M_a for each $a \in \Sigma$:

$$M_0 = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$M_1 = \begin{pmatrix} \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

We have $Q = \{q_{\text{init}}, \perp, \top\}$, $F = \{\top\}$, and, e.g., $\delta(q_{\text{init}}, 0)(\perp) = \frac{1}{2}$.

[2]Fijalkow, "Undecidability results for probabilistic automata", 2017.

Semantics

- Given a word $w = a_1 \dots a_n \in \Sigma^*$, **multiple runs** (sequences of automaton states) are possible, as in nondeterministic automata.
- With PAs, we can assign a **probability** to runs over a word.
  The **probability of a run** $\rho = q_0 q_1 \dots q_n$ **over** $w = a_1 \dots a_n$ is

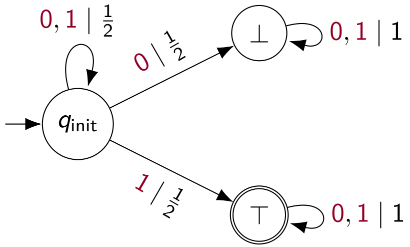
$$\mathbb{P}_{\mathcal{A}}(\rho) = \prod_{i \in \{1,\dots,n\}} \delta(q_{i-1}, a_i)(q_i),$$

  where $q_0 = q_{\mathsf{init}}$.
- A run is **accepting** if it ends in a state of F. We denote the set of accepting runs of $\mathcal{A}$ over w by $\mathsf{Runs}_{\mathcal{A}}(w)$.
- The probability of w over $\mathcal{A}$ is the sum of the probabilities of all accepting runs:

$$\mathcal{A}(w) = \sum_{\rho \in \mathsf{Runs}_{\mathcal{A}}(w)} \mathbb{P}_{\mathcal{A}}(\rho).$$

Running example

Going back to our example $\mathcal{A}_{\text{bin}}$:



- For $w = 0000$, what is $\mathcal{A}_{\text{bin}}(w)$? 0
- For $w = 110$, what is $\mathcal{A}_{\text{bin}}(w)$? $\frac{1}{2} + \frac{1}{4} = (0.11)_2$
- For $w = b_1 b_2 \dots b_n \in \{0, 1\}^n$, what is $\mathcal{A}_{\text{bin}}(w)$? $(0.b_1 b_2 \dots b_n)_2$

PA $\mathcal{A}_{\text{bin}}$ computes the function "bin" that maps a **binary word to its value** in $[0, 1]$:

$$\mathcal{A}_{\text{bin}} \colon \Sigma^* \to [0, 1]$$

$$b_1 b_2 \dots b_n \mapsto (0.b_1 b_2 \dots b_n)_2 = \sum_{i=1}^{n} \frac{b_i}{2^i}$$

Languages recognized by a PA

- As for (non)deterministic automata, we can use probabilistic automata to **define languages** $\subseteq \Sigma^*$.
- However, just accepting "words with an accepting run" is the same as for regular automata; we want to use the **probabilities**.

Languages recognized by a PA

Let $\mathcal{A}$ be a PA and pick a **threshold** $\lambda \in [0, 1]$. We define

$$\mathcal{L}_{\geq\lambda}(\mathcal{A}) = \{w \in \Sigma^* \mid \mathcal{A}(w) \geq \lambda\}.$$

For our running example $\mathcal{A}_{\text{bin}}$, we have

$$\mathcal{L}_{\geq\lambda}(\mathcal{A}_{\text{bin}}) = \{b_1 \dots b_n \in \{0, 1\}^* \mid (0.b_1 \dots b_n)_2 \geq \lambda\}.$$

Expressivity

- PAs generalize deterministic automata (using deterministic δ), so they can recognize the usual **regular languages**.
- But PAs recognize **more than just regular languages**.
  $\rightsquigarrow$ *Do you have an argument to show this?*

One possible Cantor-based argument

For automaton $\mathcal{A}_{\text{bin}}$, for $\lambda \neq \lambda' \in [0, 1]$, we have $\mathcal{L}_{\geq \lambda}(\mathcal{A}_{\text{bin}}) \neq \mathcal{L}_{\geq \lambda'}(\mathcal{A}_{\text{bin}})$.

- Hence, there are **uncountably many languages** recognized by PAs (even just by $\mathcal{A}_{\text{bin}}$).
- On the other hand, there are only **countably many (non)deterministic automata**, and thus only countably many regular languages.

Closure properties

PAs are well-behaved under usual operations.

Closure properties

Let $\mathcal{A}_1, \mathcal{A}_2$ be PAs.

- **Complementation**:
  - There is a PA computing $1 - \mathcal{A}_1$.
    $\rightsquigarrow$ Take $F' = Q \setminus F_1$.
- **Convex combinations**:
  - There is a PA computing $\frac{1}{2}\mathcal{A}_1 + \frac{1}{2}\mathcal{A}_2$.
    $\rightsquigarrow$ Take the disjoint union and add an initial state with $\frac{1}{2}$ transitions to each.
- **Products**:
  - There is a PA computing $\mathcal{A}_1 \cdot \mathcal{A}_2$.
    $\rightsquigarrow$ Take their synchronous product, with $(q_1, q_2) \in F'$ if $q_1 \in F_1$ and $q_2 \in F_2$.

Undecidability of the emptiness

When considering automata, the first decision problem that comes to mind is often the **emptiness problem**.

Emptiness problem

- **Input**: A probabilistic automaton $\mathcal{A}$ and a threshold $\lambda \in [0, 1] \cap \mathbb{Q}$.
- **Output**: **YES** if $\mathcal{L}_{\geq\lambda}(\mathcal{A}) \neq \emptyset$, **NO** otherwise.

Undecidable: Reduction from the *Post's Correspondence Problem* (PCP).
BLACKBOARD PROOF (if time allows)

Undecidability of approximation

- **Control semantics**: Assume the PA represents a computer system. We want to maximize the probability to reach a target state. We are interested in the value

$$\sup_{w \in \Sigma^*} \mathcal{A}(w).$$

- Unfortunately, no algorithm can even **approximate** this up to any $\varepsilon < \frac{1}{2}$ (proof idea: if there were, you could decide an undecidable problem).
- So most problems about PAs are **undecidable** ☹.
- In the second part of the talk, let us look at an **even more general class of models**!

Outline

Partial observations: Motivation

- In addition to control and stochasticity, it is common to add **partial observations**.
- Why? Think about **autonomous cars**. They must
  - make decisions (**control**)
  - in an uncertain environment (**modelled with stochasticity**)
  - while observing part of, but not all, their environment (**partial observations**).
- Other applications: robotics, healthcare, finance...
- Because of the "observation" part, we need a more general model.
- **Stochastic models**:

	No observation	**Partial observations**	**Perfect observations**
No control		Hidden Markov models	Markov chains
Control	PAs	**POMDPs**	MDPs

- We consider **partially observable Markov decision processes** (POMDPs), which generalize PAs and MDPs.

Tiger POMDP

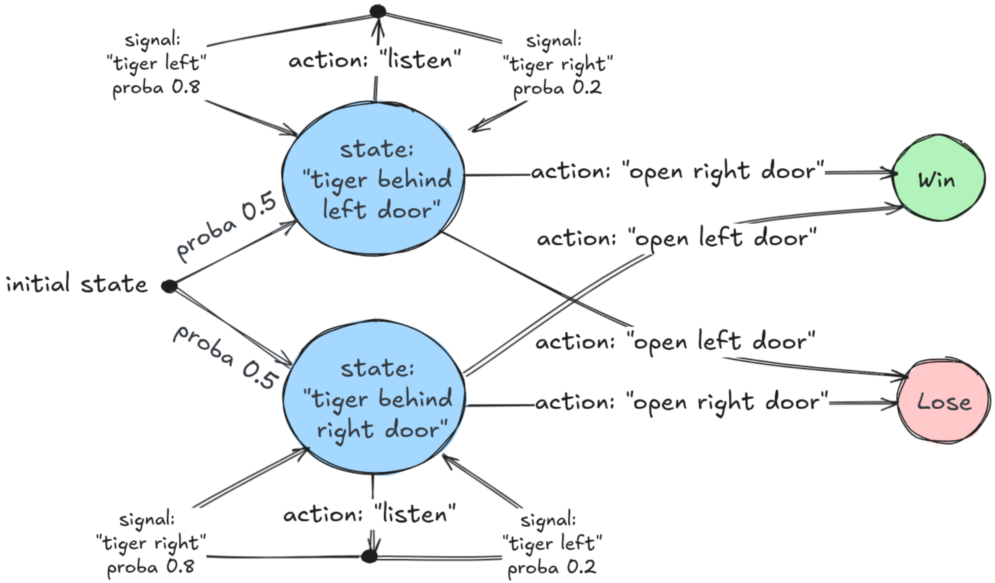
Before defining POMDPs formally, let us look at the well-known "tiger" example. . .[3]

- A person is in front of **two closed doors**.
- A tiger is behind **one** of the doors.
- They **have to open** the **non-tiger door** to win.
- They can **listen** to get some **imperfect information** about the tiger's location.

generated with ChatGPT

[3]Kaelbling, Littman, and Cassandra, "Planning and acting in partially observable stochastic domains", 1998.

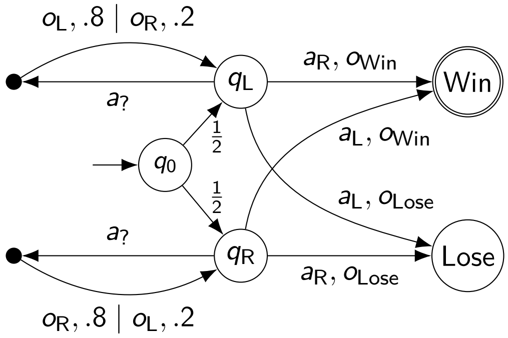
Tiger POMDP: more formally


signal: "tiger left" proba 0.8
action: "listen"
signal: "tiger right" proba 0.2
state: "tiger behind left door"
action: "open right door"
Win
action: "open left door"
proba 0.5
initial state
proba 0.5
action: "open left door"
state: "tiger behind right door"
action: "open right door"
Lose
signal: "tiger right" proba 0.8
action: "listen"
signal: "tiger left" proba 0.2

Tiger POMDP: even more formally

A POMDP is a tuple $\mathcal{P} = (Q, \mathsf{Act}, \mathsf{Obs}, q_{\mathsf{init}}, \delta, F)$ with...

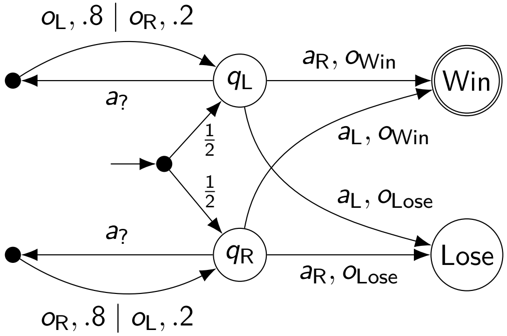
states Q, **actions** Act, **observations** Obs, **initial state** q_{init},
transitions $\delta\colon Q \times \mathsf{Act} \to \mathsf{Dist}(\mathsf{Obs} \times Q)$, and **target states** F.



$$Q = \{q_0, q_L, q_R, \mathsf{Win}, \mathsf{Lose}\},\ \mathsf{Act} = \{a_L, a_R, a_?\},\ \mathsf{Obs} = \{o_L, o_R, o_{\mathsf{Win}}, o_{\mathsf{Lose}}\},\ F = \{\mathsf{Win}\}.$$

Strategies

- We generalize the **words** from PAs: here, an agent can make decisions based on the **observations** they got.
- In POMDPs, **strategies** are functions $(\mathsf{Act} \times \mathsf{Obs})^* \to \mathsf{Dist}(\mathsf{Act})$.
- **"Maximal" probability of reaching** Win **in the tiger POMDP?**



No strategy reaches Win **with probability** 1...
However, **for every** $\varepsilon > 0$, there is a strategy that wins with probability $\geq 1 - \varepsilon$.

Learning vs. formal methods

Reinforcement learning	VS	**Formal methods**
Learns by **interacting** with the environment	Approach	Computes strategy from a **static description** of the POMDP
Model-free, online learning	Paradigm	**Model-based**, offline analysis
Easily handles **large state spaces**	Scalability	Struggles with large spaces ($\geq$ **exponential time**)
Offers **few guarantees** about the output	Reliability	Provides strict **mathematical guarantees**

Here, our approach is **formal methods**: what is **computable** about POMDPs?

What is computable about POMDPs?

Decidability in POMDPs[4,5,6,7]

- Given a POMDP and a threshold $\lambda \in (0, 1)$, is there a strategy that reaches the target with probability $\geq \lambda$? **Undecidable** 😕
- Given a POMDP and an $\varepsilon > 0$, is there an algorithm that approximates the supremum probability of reaching the target up to ε? **No** ☹
- Given a POMDP, is it true that for all $\varepsilon > 0$, there is a strategy that reaches the target with probability $\geq 1 - \varepsilon$? **Undecidable** ☹
- Given a POMDP, is there a strategy that reaches the target with probability 1? **EXPTIME-complete**! 🙂

Summary: **quantitative** problems are **all** undecidable in PAs/POMDPs.
Qualitative problems (e.g., existence of an **almost-sure strategy**): it depends!

[4] Madani, Hanks, and Condon, "On the undecidability of probabilistic planning and related stochastic optimization problems", 2003.

[5] Gimbert and Oualhadj, "Probabilistic Automata on Finite Words: Decidable and Undecidable Problems", 2010.

[6] Baier, Größer, and Bertrand, "Probabilistic ω-automata", 2012.

[7] Chatterjee, Chmelik, and Tracol, "What is decidable about partially observable Markov decision processes with ω-regular objectives", 2016.

Open problems

- **Find decidable subclasses**: Since general POMDPs are highly undecidable, research focuses on restricting the model to regain decidability.
  Multiple decidable classes are known:
  - MDPs (perfect observations),
  - deterministic POMDPs ($\delta\colon Q \times \mathsf{Act} \to \mathsf{Obs} \times Q$),[8]
  - restricting to finite-memory strategies,[9]
  - revealing POMDPs: POMDPs where the current state is known every so often.[10]
- **Long-term goal**: Cover "all" POMDPs that occur in practice. I.e., bridge the gap between theoretical decidability and the models actively used in AI and robotics.
- Study objectives **more complex than "reach some states"** (e.g., temporal constraints, quantitative rewards...) in restricted classes.

Thanks!

[8] Bonet, "Deterministic POMDPs Revisited", 2009.

[9] Chatterjee, Chmelik, and Tracol, "What is decidable about partially observable Markov decision processes with ω-regular objectives", 2016.

[10] Belly, Fijalkow, Gimbert, Horn, Pérez, and Vandenhove, "Revelations: A Decidable Class of POMDPs with Omega-Regular Objectives", 2025.

Not covered due to a lack of time

Beyond immediate observations? Beliefs!

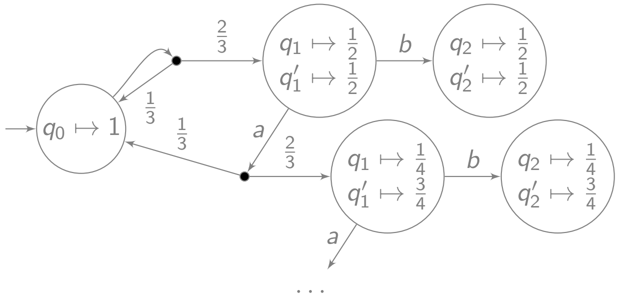
- In POMDPs, the agent does not know the actual state, but they can **keep track of their belief** about the current state.
- A **belief** is a **probability distribution over states**.
- From a current belief $b \in \mathrm{Dist}(Q)$, say we play a and receive observation o.
  Then, we believe we are in q' with probability (Bayes' theorem)...

$$b'(q') = \frac{\delta(o \mid q', a) \sum_{q \in Q} \delta(q' \mid q, a) b(q)}{\sum_{q'' \in Q} \delta(o \mid q'', a) \sum_{q \in Q} \delta(q'' \mid q, a) b(q)}.$$

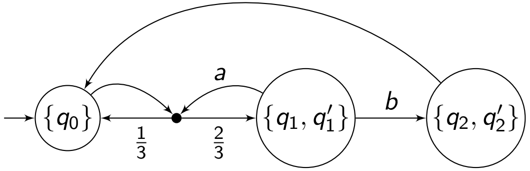
- Not "nice": less obvious **algebraic structure** than, e.g., in a Markov chain or a graph (where updates are just matrix multiplication).

Belief (support) MDP

POMDPs induce
infinite belief MDPs.
Source of undecidability!


$q_0 \mapsto 1$
$\frac{2}{3}$
$\frac{1}{3}$
$\frac{1}{3}$
$q_1 \mapsto \frac{1}{2}$, $q_1' \mapsto \frac{1}{2}$
b
$q_2 \mapsto \frac{1}{2}$, $q_2' \mapsto \frac{1}{2}$
a
$\frac{2}{3}$
$q_1 \mapsto \frac{1}{4}$, $q_1' \mapsto \frac{3}{4}$
b
$q_2 \mapsto \frac{1}{4}$, $q_2' \mapsto \frac{3}{4}$
a
$\dots$

Finite MDP: only keep belief **supports**.



When does the analysis of the belief **support** MDP suffice?

In general, neither sound nor complete. . .