Controllers in Reactive Synthesis: A Strategic Perspective

Mickael Randour

F.R.S.-FNRS & UMONS – Université de Mons, Belgium

November 21, 2024

Journées annuelles du GT Vérif 2024

The talk in one slide

The talk in one slide

Strategies = **formal blueprints** for real-world controllers.

The talk in one slide

Strategies = **formal blueprints** for real-world controllers.

Simpler is better:

- easier to understand,
- cheaper to produce and maintain.

The talk in one slide

Strategies = **formal blueprints** for real-world controllers.

Simpler is better:

- ▷ easier to understand,
- ▷ cheaper to produce and maintain.

Aim of this survey talk

Understanding **how complex** strategies need to be.

The talk in one slide

Strategies = **formal blueprints** for real-world controllers.

Simpler is better:

- ▷ easier to understand,
- ▷ cheaper to produce and maintain.

Aim of this survey talk

Understanding **how complex** strategies need to be.

But how to define complexity and how to measure it?

The talk in one slide

Strategies = **formal blueprints** for real-world controllers.

Simpler is better:

- ▷ easier to understand,
- ▷ cheaper to produce and maintain.

Aim of this survey talk

Understanding **how complex** strategies need to be.

But how to define complexity and how to measure it?

$\hookrightarrow$ **That is our topic of the today.**

The talk in one **more** slide

The talk in one **more** slide

Yes, I lied, and I will lie even more. The results I will survey span numerous combinations of

- game models,
- strategy models,
- objectives,
- decision problems...

The talk in one **more** slide

Yes, I lied, and I will lie even more. The results I will survey span numerous combinations of

- game models,
- strategy models,
- objectives,
- decision problems...

There will be some hand-waving and approximations to keep the talk high level.

The talk in one **more** slide

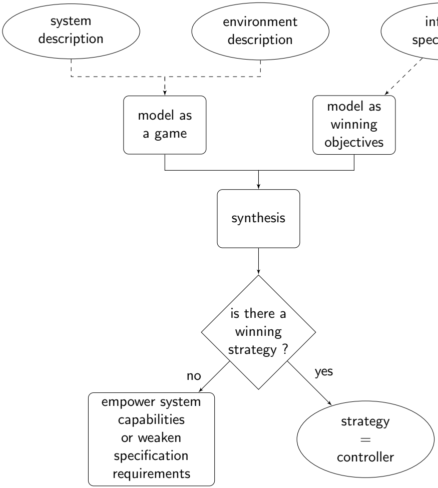
Yes, I lied, and I will lie even more. The results I will survey span numerous combinations of

- game models,
- strategy models,
- objectives,
- decision problems...

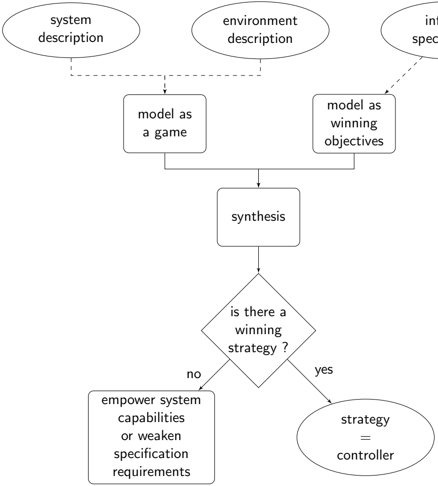
There will be some hand-waving and approximations to keep the talk high level.

$\hookrightarrow$ **I will focus on recent work with marvelous co-authors.**

Controller synthesis: a game-theoretic approach



Controller synthesis: a game-theoretic approach

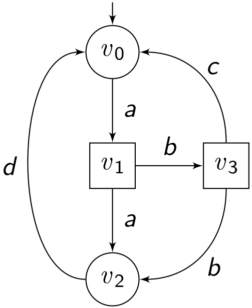


A plethora of models and objectives exist.[1]

Our focus here: how complex **strategies** need to be?

[1] Randour, "Automated Synthesis of Reliable and Efficient Systems Through Game Theory: A Case Study", 2013; Clarke et al., Handbook of Model Checking, 2018; Fijalkow et al., Games on Graphs, 2023.

Two-player games



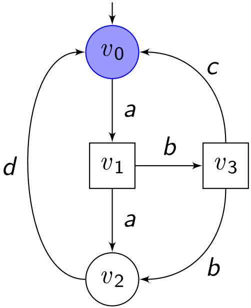
A two-player turn-based finite **arena** $\mathcal{A} = (V_{\bigcirc}, V_{\Box}, E)$ with no deadlock.

Color function $\mathfrak{c} \colon E \to C$.

$\hookrightarrow$ Players move a pebble along the edges creating an infinite **play**.

$\hookrightarrow$ **Behavior of the system = sequence of colors.**

Two-player games



A two-player turn-based finite **arena** $\mathcal{A} = (V_\bigcirc, V_\square, E)$ with no deadlock.

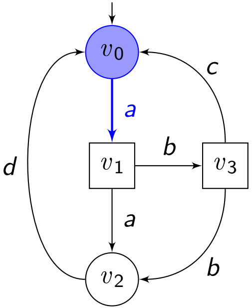
Color function $\mathfrak{c}\colon E \to C$.

$\hookrightarrow$ Players move a pebble along the edges creating an infinite **play**.

$\hookrightarrow$ **Behavior of the system = sequence of colors.**

Sample play:

Two-player games



A two-player turn-based finite **arena** $\mathcal{A} = (V_{\bigcirc}, V_{\square}, E)$ with no deadlock.

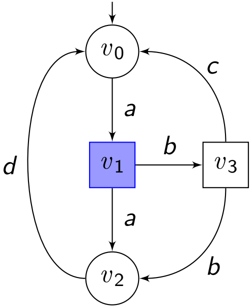
Color function $\mathfrak{c}\colon E \to C$.

$\hookrightarrow$ Players move a pebble along the edges creating an infinite **play**.

$\hookrightarrow$ **Behavior of the system = sequence of colors.**

Sample play: a

Two-player games



A two-player turn-based finite **arena** $\mathcal{A} = (V_{\bigcirc}, V_{\square}, E)$ with no deadlock.

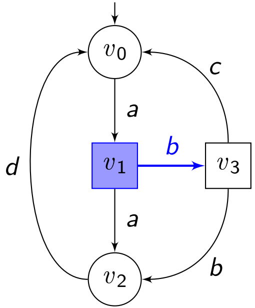
Color function $\mathfrak{c}\colon E \to C$.

$\hookrightarrow$ Players move a pebble along the edges creating an infinite **play**.

$\hookrightarrow$ **Behavior of the system = sequence of colors.**

Sample play: a

Two-player games



A two-player turn-based finite **arena** $\mathcal{A} = (V_{\bigcirc}, V_{\Box}, E)$ with no deadlock.

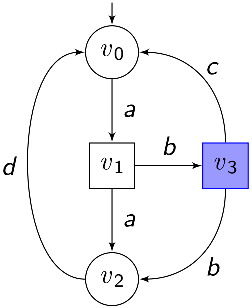
Color function $\mathfrak{c}\colon E \to C$.

$\hookrightarrow$ Players move a pebble along the edges creating an infinite **play**.

$\hookrightarrow$ **Behavior of the system = sequence of colors.**

Sample play: ab

Two-player games



A two-player turn-based finite **arena** $\mathcal{A} = (V_\bigcirc, V_\Box, E)$ with no deadlock.

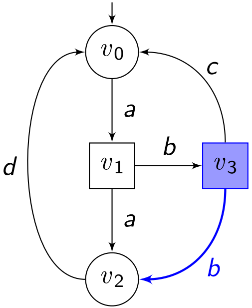
Color function $\mathfrak{c}\colon E \to C$.

$\hookrightarrow$ Players move a pebble along the edges creating an infinite **play**.

$\hookrightarrow$ **Behavior of the system = sequence of colors.**

Sample play: ab

Two-player games



A two-player turn-based finite **arena** $\mathcal{A} = (V_{\bigcirc}, V_{\Box}, E)$ with no deadlock.

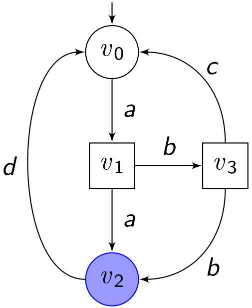
Color function $\mathfrak{c} \colon E \to C$.

$\hookrightarrow$ Players move a pebble along the edges creating an infinite **play**.

$\hookrightarrow$ **Behavior of the system = sequence of colors.**

Sample play: *abb*

Two-player games



A two-player turn-based finite **arena** $\mathcal{A} = (V_{\bigcirc}, V_{\Box}, E)$ with no deadlock.

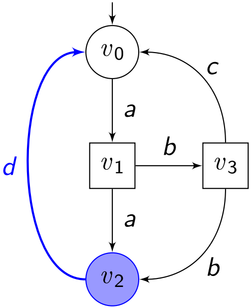
Color function $\mathfrak{c} \colon E \to C$.

$\hookrightarrow$ Players move a pebble along the edges creating an infinite **play**.

$\hookrightarrow$ **Behavior of the system = sequence of colors.**

Sample play: *abb*

Two-player games



A two-player turn-based finite **arena** $\mathcal{A} = (V_{\bigcirc}, V_{\Box}, E)$ with no deadlock.

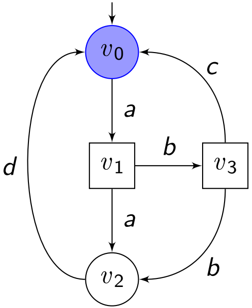
Color function $\mathfrak{c} \colon E \to C$.

$\hookrightarrow$ Players move a pebble along the edges creating an infinite **play**.

$\hookrightarrow$ **Behavior of the system = sequence of colors.**

Sample play: *abbd*

Two-player games



A two-player turn-based finite **arena** $\mathcal{A} = (V_{\bigcirc}, V_{\Box}, E)$ with no deadlock.

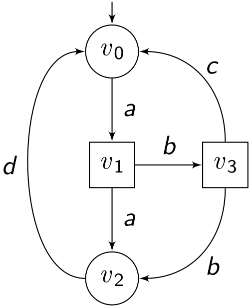
Color function $\mathfrak{c}\colon E \to C$.

$\hookrightarrow$ Players move a pebble along the edges creating an infinite **play**.

$\hookrightarrow$ **Behavior of the system = sequence of colors.**

Sample play: *abbd*

Two-player games



A two-player turn-based finite **arena** $\mathcal{A} = (V_{\bigcirc}, V_{\Box}, E)$ with no deadlock.

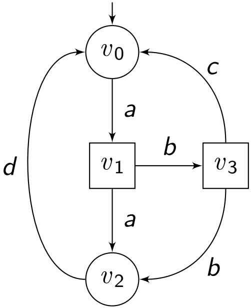
Color function $\mathfrak{c}\colon E \to C$.

$\hookrightarrow$ Players move a pebble along the edges creating an infinite **play**.

$\hookrightarrow$ **Behavior of the system = sequence of colors.**

Sample play: $abbd \ldots \in C^\omega$

Two-player games



A two-player turn-based finite **arena** $\mathcal{A} = (V_{\bigcirc}, V_{\Box}, E)$ with no deadlock.

Color function $\mathfrak{c}\colon E \to C$.

$\hookrightarrow$ Players move a pebble along the edges creating an infinite **play**.

$\hookrightarrow$ **Behavior of the system = sequence of colors.**

Usual interpretation

$\mathcal{P}_{\bigcirc}$ (the system to control) tries to satisfy its **specification** while $\mathcal{P}_{\Box}$ (the environment) tries to prevent it from doing so.

Specifications

They are encoded as some kind of *objective* defined using colors.
Three main flavors:

Specifications

They are encoded as some kind of *objective* defined using colors.
Three main flavors:

1. A **winning condition**: a set of winning plays that $\mathcal{P}_{\bigcirc}$ tries to realize. E.g., $\mathsf{Reach}(t) = \{\pi = c_0 c_1 c_2 \ldots \mid t \in \pi\}$, for $t \in C$ a given color, a *reachability* objective.

Specifications

They are encoded as some kind of *objective* defined using colors.
Three main flavors:

1. A **winning condition**: a set of winning plays that $\mathcal{P}_{\bigcirc}$ tries to realize. E.g., $\mathsf{Reach}(t) = \{\pi = c_0 c_1 c_2 \ldots \mid t \in \pi\}$, for $t \in C$ a given color, a *reachability* objective.
2. A **payoff function** to optimize, assuming $C \subset \mathbb{Q}$. E.g., the *discounted sum* function, defined as $\mathsf{DS}(\pi) = \sum_{i=0}^{\infty} \gamma^i c_i$ for some discount factor $\gamma \in]0, 1[$.

Specifications

They are encoded as some kind of *objective* defined using colors.
Three main flavors:

1. A **winning condition**: a set of winning plays that $\mathcal{P}_{\bigcirc}$ tries to realize. E.g., $\mathsf{Reach}(t) = \{\pi = c_0 c_1 c_2 \ldots \mid t \in \pi\}$, for $t \in C$ a given color, a *reachability* objective.
2. A **payoff function** to optimize, assuming $C \subset \mathbb{Q}$. E.g., the *discounted sum* function, defined as $\mathsf{DS}(\pi) = \sum_{i=0}^{\infty} \gamma^i c_i$ for some discount factor $\gamma \in]0, 1[$.
3. A **preference relation** defines a total preorder over sequences of colors, thus generalizing both previous concepts.

Strategies

Player $\mathcal{P}_\nabla$ chooses outgoing edges following a **strategy**

$$\sigma_\nabla \colon V^* V_\nabla \to V$$

consistent with the underlying graph.

Strategies

Player $\mathcal{P}_\nabla$ chooses outgoing edges following a **strategy**

$$\sigma_\nabla \colon V^* V_\nabla \to V$$

consistent with the underlying graph.

$\hookrightarrow$ We are interested in the complexity of **optimal strategies.**

Strategies

Player $\mathcal{P}_\triangledown$ chooses outgoing edges following a **strategy**

$$\sigma_\triangledown \colon V^* V_\triangledown \to V$$

consistent with the underlying graph.

$\hookrightarrow$ We are interested in the complexity of **optimal strategies.**

Optimal strategies (using a preference relation $\sqsubseteq$)

A strategy $\sigma_\bigcirc$ of $\mathcal{P}_\bigcirc$ is optimal if it guarantees (i.e., against an optimal adversary $\mathcal{P}_\square$) a play at least as good as any other strategy $\sigma'_\bigcirc$ with respect to $\sqsubseteq$.

MDPs & stochastic games

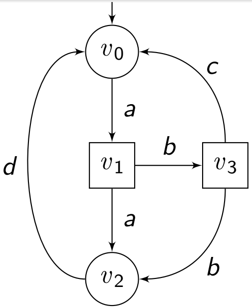
Why?

In many real scenarios, the environment is not fully antagonistic, but exhibits stochastic behaviors.

MDPs & stochastic games

Why?

In many real scenarios, the environment is not fully antagonistic, but exhibits stochastic behaviors.



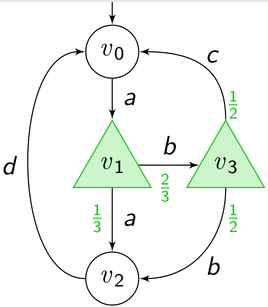
Two-player (deterministic) game.

$$V = V_{\bigcirc} \uplus V_{\Box}.$$

MDPs & stochastic games

Why?

In many real scenarios, the environment is not fully antagonistic, but exhibits stochastic behaviors.



Markov decision process.

$$V = V_{\bigcirc} \uplus V_{\triangle}.$$

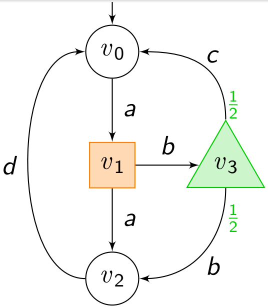
Either $\mathcal{P}_{\bigcirc}$ aims to maximize

- ▷ $\mathbb{P}^{\sigma_{\bigcirc}}[W]$ for some winning condition W,
- ▷ or $\mathbb{E}^{\sigma_{\bigcirc}}[f]$ for some payoff function f.

MDPs & stochastic games

Why?

In many real scenarios, the environment is not fully antagonistic, but exhibits stochastic behaviors.



Stochastic game.

$$V = V_{\bigcirc} \uplus V_{\triangle} \uplus V_{\square}.$$

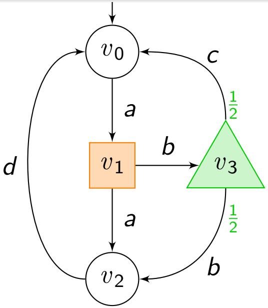
Either $\mathcal{P}_{\bigcirc}$ aims to maximize, against the adversary $\mathcal{P}_{\square}$,

- $\mathbb{P}^{\sigma_{\bigcirc}, \sigma_{\square}}[W]$ for some winning condition W,
- or $\mathbb{E}^{\sigma_{\bigcirc}, \sigma_{\square}}[f]$ for some payoff function f.

MDPs & stochastic games

Why?

In many real scenarios, the environment is not fully antagonistic, but exhibits stochastic behaviors.



Stochastic game.

$$V = V_\bigcirc \uplus V_\triangle \uplus V_\Box.$$

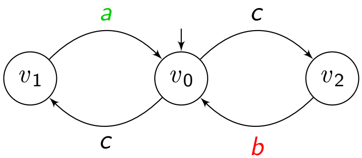
Actions

We often use **actions** instead of stochastic vertices.

Multiple objectives

Combining objectives

Complex objectives arise when combining simple objectives, and usually require more complex strategies to play optimally.

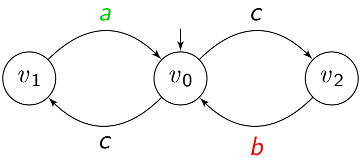


Seeing *a* **and** *b* infinitely often requires memory, but seeing only one does not (Büchi objective).

Multiple objectives

Combining objectives

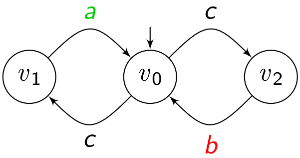
Complex objectives arise when combining simple objectives, and usually require more complex strategies to play optimally.

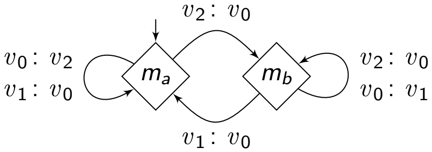


Seeing a **and** b infinitely often requires memory, but seeing only one does not (Büchi objective).

$\hookrightarrow$ We are often interested in the **Pareto front**, i.e., all payoff vectors not dominated by another.

Classical representation of strategies: Mealy machines



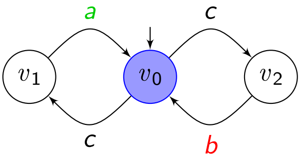


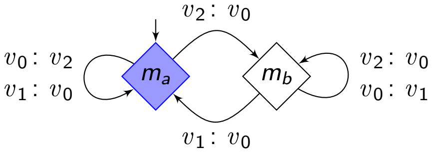
Mealy machine $\mathcal{M} = \{M, m_{\mathsf{init}}, \alpha_{\mathsf{nxt}}, \alpha_{\mathsf{up}}\}$:

- M is the set of *memory states*,
- m_{init} is the *initial state*,
- $\alpha_{\mathsf{nxt}} \colon M \times V \to V$ is the *next-action function*,
- $\alpha_{\mathsf{up}} \colon M \times V \to M$ is the *update function*.

Finite memory if $|M| < \infty$, memoryless if $|M| = 1$.

Classical representation of strategies: Mealy machines



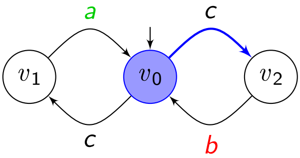


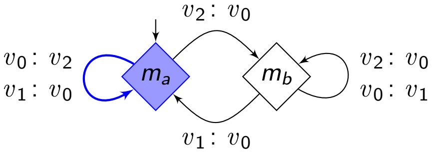
Mealy machine $\mathcal{M} = \{M, m_{\text{init}}, \alpha_{\text{nxt}}, \alpha_{\text{up}}\}$:

- M is the set of *memory states*,
- m_{init} is the *initial state*,
- $\alpha_{\text{nxt}} \colon M \times V \to V$ is the *next-action function*,
- $\alpha_{\text{up}} \colon M \times V \to M$ is the *update function*.

Finite memory if $|M| < \infty$, memoryless if $|M| = 1$.

Classical representation of strategies: Mealy machines



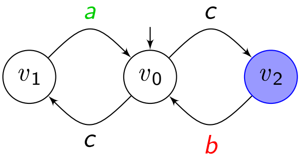


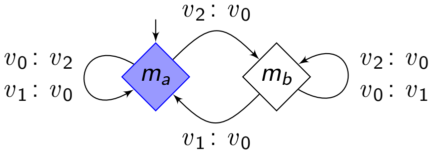
Mealy machine $\mathcal{M} = \{M, m_{\text{init}}, \alpha_{\text{nxt}}, \alpha_{\text{up}}\}$:

- M is the set of *memory states*,
- m_{init} is the *initial state*,
- $\alpha_{\text{nxt}} \colon M \times V \to V$ is the *next-action function*,
- $\alpha_{\text{up}} \colon M \times V \to M$ is the *update function*.

Finite memory if $|M| < \infty$, memoryless if $|M| = 1$.

Classical representation of strategies: Mealy machines



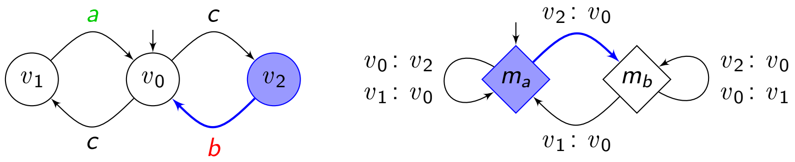


Mealy machine $\mathcal{M} = \{M, m_{\text{init}}, \alpha_{\text{nxt}}, \alpha_{\text{up}}\}$:

- M is the set of *memory states*,
- m_{init} is the *initial state*,
- $\alpha_{\text{nxt}} : M \times V \to V$ is the *next-action function*,
- $\alpha_{\text{up}} : M \times V \to M$ is the *update function*.

Finite memory if $|M| < \infty$, memoryless if $|M| = 1$.

Classical representation of strategies: Mealy machines

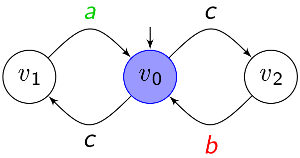


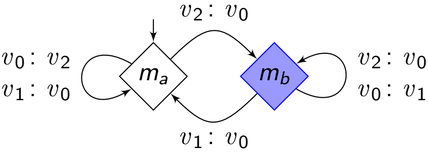
Mealy machine $\mathcal{M} = \{M, m_{\text{init}}, \alpha_{\text{nxt}}, \alpha_{\text{up}}\}$:

- M is the set of *memory states*,
- m_{init} is the *initial state*,
- $\alpha_{\text{nxt}} : M \times V \to V$ is the *next-action function*,
- $\alpha_{\text{up}} : M \times V \to M$ is the *update function*.

Finite memory if $|M| < \infty$, memoryless if $|M| = 1$.

Classical representation of strategies: Mealy machines



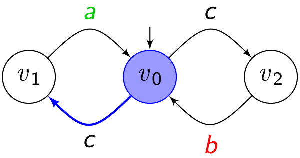


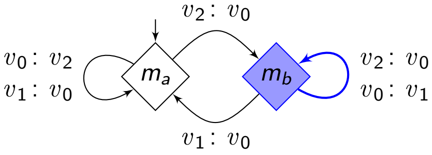
Mealy machine $\mathcal{M} = \{M, m_{\mathsf{init}}, \alpha_{\mathsf{nxt}}, \alpha_{\mathsf{up}}\}$:

- M is the set of *memory states*,
- m_{init} is the *initial state*,
- $\alpha_{\mathsf{nxt}} \colon M \times V \to V$ is the *next-action function*,
- $\alpha_{\mathsf{up}} \colon M \times V \to M$ is the *update function*.

Finite memory if $|M| < \infty$, memoryless if $|M| = 1$.

Classical representation of strategies: Mealy machines



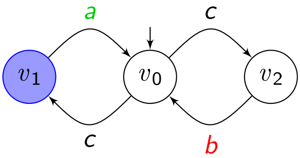


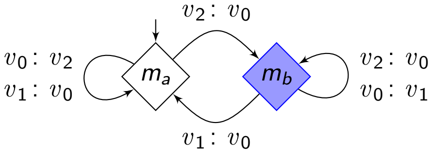
Mealy machine $\mathcal{M} = \{M, m_{\text{init}}, \alpha_{\text{nxt}}, \alpha_{\text{up}}\}$:

- M is the set of *memory states*,
- m_{init} is the *initial state*,
- $\alpha_{\text{nxt}}\colon M \times V \to V$ is the *next-action function*,
- $\alpha_{\text{up}}\colon M \times V \to M$ is the *update function*.

Finite memory if $|M| < \infty$, memoryless if $|M| = 1$.

Classical representation of strategies: Mealy machines



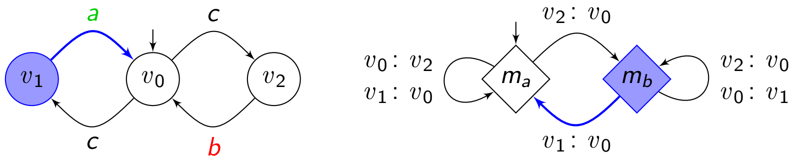


Mealy machine $\mathcal{M} = \{M, m_{\text{init}}, \alpha_{\text{nxt}}, \alpha_{\text{up}}\}$:

- M is the set of *memory states*,
- m_{init} is the *initial state*,
- $\alpha_{\text{nxt}} \colon M \times V \to V$ is the *next-action function*,
- $\alpha_{\text{up}} \colon M \times V \to M$ is the *update function*.

Finite memory if $|M| < \infty$, memoryless if $|M| = 1$.

Classical representation of strategies: Mealy machines

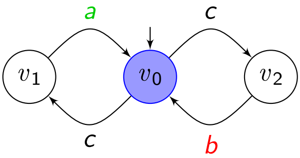


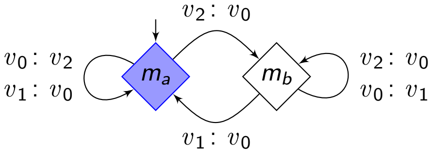
Mealy machine $\mathcal{M} = \{M, m_{\mathsf{init}}, \alpha_{\mathsf{nxt}}, \alpha_{\mathsf{up}}\}$:

- M is the set of *memory states*,
- m_{init} is the *initial state*,
- $\alpha_{\mathsf{nxt}} \colon M \times V \to V$ is the *next-action function*,
- $\alpha_{\mathsf{up}} \colon M \times V \to M$ is the *update function*.

Finite memory if $|M| < \infty$, memoryless if $|M| = 1$.

Classical representation of strategies: Mealy machines



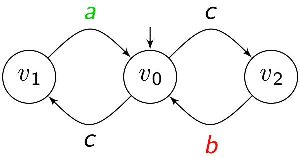


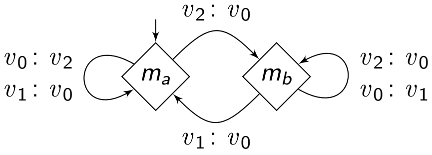
Mealy machine $\mathcal{M} = \{M, m_{\text{init}}, \alpha_{\text{nxt}}, \alpha_{\text{up}}\}$:

- M is the set of *memory states*,
- m_{init} is the *initial state*,
- $\alpha_{\text{nxt}} \colon M \times V \to V$ is the *next-action function*,
- $\alpha_{\text{up}} \colon M \times V \to M$ is the *update function*.

Finite memory if $|M| < \infty$, memoryless if $|M| = 1$.

Classical representation of strategies: Mealy machines



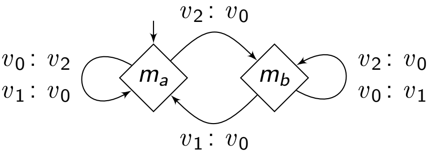


Mealy machine $\mathcal{M} = \{M, m_{\text{init}}, \alpha_{\text{nxt}}, \alpha_{\text{up}}\}$:

- M is the set of *memory states*,
- m_{init} is the *initial state*,
- $\alpha_{\text{nxt}} \colon M \times V \to V$ is the *next-action function*,
- $\alpha_{\text{up}} \colon M \times V \to M$ is the *update function*.

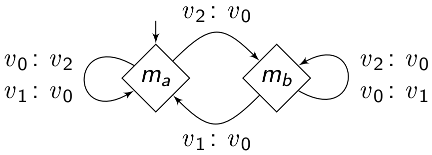
Finite memory if $|M| < \infty$, memoryless if $|M| = 1$.

The ice cream conundrum



This Mealy machine uses **chaotic** (or general) memory: it looks at the actual vertices of the game to update its memory.

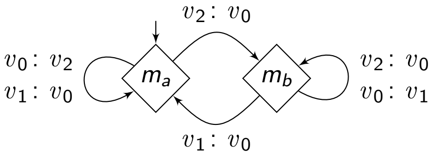
The ice cream conundrum



This Mealy machine uses **chaotic** (or general) memory: it looks at the actual vertices of the game to update its memory.

Many other flavors exist: **chromatic** memory, with or without **ε-transitions**, with different types of **randomness**, etc.

The ice cream conundrum



This Mealy machine uses **chaotic** (or general) memory: it looks at the actual vertices of the game to update its memory.

Many other flavors exist: **chromatic** memory, with or without **ε-transitions**, with different types of **randomness**, etc.

$\hookrightarrow$ **We will discuss some of these.**

Some amazing co-authors




Section mostly based on joint work with Patricia Bouyer, Stéphane Le Roux, Youssouf Oualhadj, and Pierre Vandenhove.[2]

[2] Bouyer, Le Roux, et al., "Games Where You Can Play Optimally with Arena-Independent Finite Memory", 2022; Bouyer, Oualhadj, et al., "Arena-Independent Finite-Memory Determinacy in Stochastic Games", 2023; Bouyer, Randour, and Vandenhove, "Characterizing Omega-Regularity through Finite-Memory Determinacy of Games on Infinite Graphs", 2023.

Memoryless strategies

Functions $\sigma_{\triangledown} \colon V_{\triangledown} \to V$.

- Equivalently, Mealy machines with one state.
- **Arguably**, the simplest kind of strategies.

Memoryless strategies

Functions $\sigma_{\triangledown} \colon V_{\triangledown} \to V$.

- Equivalently, Mealy machines with one state.
- **Arguably**, the simplest kind of strategies.
- Sufficient to play optimally for most *single* objectives in (stochastic) games: reachability, parity, mean-payoff, discounted sum, etc.

Starting point of our journey: *deterministic* games

Gimbert and Zielonka's characterization[3]

Memoryless strategies suffice (for both players) for a preference relation $\sqsubseteq$ iff it is **monotone** and **selective**.

[3] Gimbert and Zielonka, "Games Where You Can Play Optimally Without Any Memory", 2005.

Starting point of our journey: *deterministic* games

Gimbert and Zielonka's characterization[3]

Memoryless strategies suffice (for both players) for a preference relation $\sqsubseteq$ iff it is **monotone** and **selective**.

Corollary: one-to-two-player lift

If $\sqsubseteq$ is such that

1. in all $\mathcal{P}_{\bigcirc}$-arenas, $\mathcal{P}_{\bigcirc}$ has optimal memoryless strategies,
2. in all $\mathcal{P}_{\Box}$-arenas, $\mathcal{P}_{\Box}$ has optimal memoryless strategies,

then **both** players have optimal memoryless strategies in all **two-player** arenas.

$\Longrightarrow$ **Extremely useful as analyzing one-player games (i.e., graphs) is much easier.**

[3] Gimbert and Zielonka, "Games Where You Can Play Optimally Without Any Memory", 2005.

Handling finite-memory strategies (1/3)

Why?

- More complex objectives may require **finite** (multi-Büchi) or **infinite** memory (multi-mean-payoff).

Handling finite-memory strategies (1/3)

Why?

- More complex objectives may require **finite** (multi-Büchi) or **infinite** memory (multi-mean-payoff).

$\hookrightarrow$ **One would hope for an equivalent of Gimbert and Zielonka's result for finite memory.**

Handling finite-memory strategies (1/3)

Why?

- ▷ More complex objectives may require **finite** (multi-Büchi) or **infinite** memory (multi-mean-payoff).

$\hookrightarrow$ **One would hope for an equivalent of Gimbert and Zielonka's result for finite memory.**

Unfortunately, it does not hold.

Handling finite-memory strategies (2/3)

Let $C \subseteq \mathbb{Z}$ and the winning condition for $\mathcal{P}_{\bigcirc}$ be

$$\overline{TP}(\pi) = \infty \quad \vee \quad \exists^{\infty} n \in \mathbb{N}, \sum_{i=0}^{n} c_i = 0$$

Handling finite-memory strategies (2/3)

Let $C \subseteq \mathbb{Z}$ and the winning condition for $\mathcal{P}_{\bigcirc}$ be

$$\overline{TP}(\pi) = \infty \quad \vee \quad \exists^{\infty} n \in \mathbb{N}, \sum_{i=0}^{n} c_i = 0$$

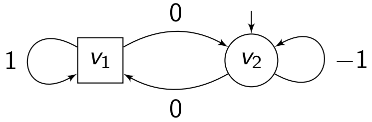
Both one-player variants are finite-memory determined.

Handling finite-memory strategies (2/3)

Let $C \subseteq \mathbb{Z}$ and the winning condition for $\mathcal{P}_{\bigcirc}$ be

$$\overline{TP}(\pi) = \infty \quad \vee \quad \exists^{\infty} n \in \mathbb{N}, \sum_{i=0}^{n} c_i = 0$$

Both one-player variants are finite-memory determined.



But the two-player one is not!
$\Longrightarrow$ $\mathcal{P}_{\bigcirc}$ **needs infinite memory to win.**

Handling finite-memory strategies (3/3)

A new frontier

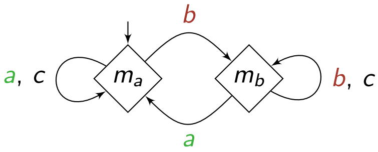
We focus on **arena-independent chromatic** memory structures.

Handling finite-memory strategies (3/3)

A new frontier

We focus on **arena-independent chromatic** memory structures.

Example for $C = \{a, b, c\}$ and objective Büchi(a) $\cap$ Büchi(b).

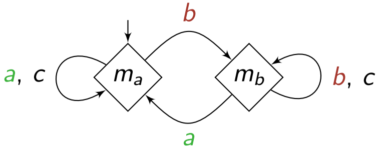


Handling finite-memory strategies (3/3)

A new frontier

We focus on **arena-independent chromatic** memory structures.

Example for $C = \{a, b, c\}$ and objective Büchi(a) $\cap$ Büchi(b).



This memory structure **suffices in all arenas**, i.e., it is always possible to find a suitable α_{nxt} to build an optimal Mealy machine.

Handling finite-memory strategies (3/3)

A new frontier

We focus on **arena-independent chromatic** memory structures.

Our characterization[4]

We obtain an equivalent to Gimbert and Zielonka's for finite memory:

1. a characterization through the concepts of $\mathcal{M}$-monotony and $\mathcal{M}$-selectivity,
2. a **one-to-two-player lift**.

[4] Bouyer, Le Roux, et al., "Games Where You Can Play Optimally with Arena-Independent Finite Memory", 2022.

Extension to stochastic games

We lift[5] this result to **pure arena-independent finite-memory** strategies in **stochastic games**:

1. characterization based on generalizations of $\mathcal{M}$-monotony and $\mathcal{M}$-selectivity,
2. **one-to-two-player lift**, from MDPs to stochastic games.

[5] Bouyer, Oualhadj, et al., "Arena-Independent Finite-Memory Determinacy in Stochastic Games", 2023.

Extension to infinite (deterministic) arenas (1/2)

We consider arenas of arbitrary cardinality and allow infinite branching.

Observation

Memory requirements can be **higher in infinite arenas**: e.g., mean-payoff objectives require infinite memory.

Extension to infinite (deterministic) arenas (1/2)

We consider arenas of arbitrary cardinality and allow infinite branching.

Observation

Memory requirements can be **higher in infinite arenas**: e.g., mean-payoff objectives require infinite memory.

The case of ω-regular objectives[6]

If a victory condition W is ω-regular, **then** it admits finite-memory optimal strategies in all (infinite) arenas.

[6] Mostowski, "Regular expressions for infinite trees and a standard form of automata", 1985; W. Zielonka, "Infinite games on finitely coloured graphs with applications to automata on infinite trees", 1998.

Extension to infinite (deterministic) arenas (2/2)

The converse[7]

If a **chromatic finite-memory** structure $\mathcal{M}$ suffices for W in all infinite arenas, **then** W is ω-regular.

$\hookrightarrow$ We build a parity automaton for W, based on $\mathcal{M}$ and $\mathcal{S}_W$, the *prefix-classifier* of W (recognizing its Myhill-Nerode classes).

[7] Bouyer, Randour, and Vandenhove, "Characterizing Omega-Regularity through Finite-Memory Determinacy of Games on Infinite Graphs", 2023.

Extension to infinite (deterministic) arenas (2/2)

The converse[7]

If a **chromatic finite-memory** structure $\mathcal{M}$ suffices for W in all infinite arenas, **then** W is ω-regular.

$\hookrightarrow$ We build a parity automaton for W, based on $\mathcal{M}$ and $\mathcal{S}_W$, the *prefix-classifier* of W (recognizing its Myhill-Nerode classes).

Corollaries

1. Game-theoretical characterization of ω-regularity.
2. **One-to-two-player lift** for infinite arenas.

[7] Bouyer, Randour, and Vandenhove, "Characterizing Omega-Regularity through Finite-Memory Determinacy of Games on Infinite Graphs", 2023.

Other criteria and characterizations

There is a plethora of results related to memory (models vary). Non-exhaustive list:

- characterizations through universal graphs,[8]
- tight memory bounds for sub-classes of objectives,[9]
- criteria for half-positionality,[10]
- one-to-multi-objective lift,[11]
- two-to-multi-player lift.[12]

$\hookrightarrow$ Find more about **chromatic memory** in our survey.[13]

---

[8] Casares and Ohlmann, "Characterising Memory in Infinite Games", 2023.

[9] Bouyer, Casares, et al., "Half-Positional Objectives Recognized by Deterministic Büchi Automata", 2024; Bouyer, Fijalkow, et al., "How to Play Optimally for Regular Objectives?", 2023; Casares and Ohlmann, "Positional ω-regular languages", 2024.

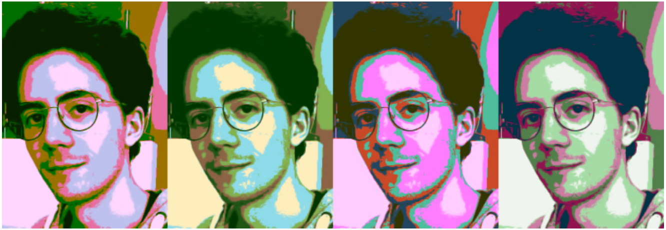
[10] Aminof and Rubin, "First-cycle games", 2017.

[11] Le Roux, Pauly, and Randour, "Extending Finite-Memory Determinacy by Boolean Combination of Winning Conditions", 2018.

[12] Le Roux and Pauly, "Extending Finite Memory Determinacy to Multiplayer Games", 2016.

[13] Bouyer, Randour, and Vandenhove, "The True Colors of Memory: A Tour of Chromatic-Memory Strategies in Zero-Sum Games on Graphs (Invited Talk)", 2022.

The amazing Mr. Main



Section mostly based on (ongoing) joint work with James C. A. Main.[14]

[14] Main and Randour, "Different Strokes in Randomised Strategies: Revisiting Kuhn's Theorem Under Finite-Memory Assumptions", 2024.

Introducing randomness in strategies (1/2)

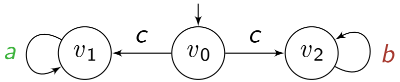
A **pure** strategy is a function $\sigma_{\triangledown} \colon V^* V_{\triangledown} \to V$.

Introducing randomness in strategies (1/2)

A **pure** strategy is a function $\sigma_{\triangledown} \colon V^* V_{\triangledown} \to V$.

We may need **randomness** to deal with, e.g.,

- multiple objectives,
- concurrent games,
- imperfect information.



Objective: $\mathbb{P}^{\sigma_{\bigcirc}}[\mathsf{Reach}(a)] \geq \frac{1}{2} \wedge \mathbb{P}^{\sigma_{\bigcirc}}[\mathsf{Reach}(b)] \geq \frac{1}{2}$

$\hookrightarrow$ **Achievable by tossing a coin in v_0.**

Introducing randomness in strategies (2/2)

Several ways of **randomizing** $\sigma_{\nabla} \colon V^{*}V_{\nabla} \rightarrow V$:

Introducing randomness in strategies (2/2)

Several ways of **randomizing** $\sigma_{\triangledown}\colon V^{*}V_{\triangledown} \rightarrow V$:

Behavioral strategies
$\sigma_{\triangledown}\colon V^{*}V_{\triangledown} \rightarrow \mathcal{D}(V)$

Introducing randomness in strategies (2/2)

Several ways of **randomizing** $\sigma_\triangledown \colon V^* V_\triangledown \to V$:

Behavioral strategies

$\sigma_\triangledown \colon V^* V_\triangledown \to \mathcal{D}(V)$

Mixed strategies

$\mathcal{D}(\sigma_\triangledown \colon V^* V_\triangledown \to V)$

Introducing randomness in strategies (2/2)

Several ways of **randomizing** $\sigma_{\nabla}\colon V^{*}V_{\nabla} \to V$:

Behavioral strategies
$\sigma_{\nabla}\colon V^{*}V_{\nabla} \to \mathcal{D}(V)$

Mixed strategies
$\mathcal{D}(\sigma_{\nabla}\colon V^{*}V_{\nabla} \to V)$

General strategies
$\mathcal{D}(\sigma_{\nabla}\colon V^{*}V_{\nabla} \to \mathcal{D}(V))$

Introducing randomness in strategies (2/2)

Several ways of **randomizing** $\sigma_\nabla \colon V^* V_\nabla \to V$:

Behavioral strategies
$\sigma_\nabla \colon V^* V_\nabla \to \mathcal{D}(V)$

Mixed strategies
$\mathcal{D}(\sigma_\nabla \colon V^* V_\nabla \to V)$

General strategies
$\mathcal{D}(\sigma_\nabla \colon V^* V_\nabla \to \mathcal{D}(V))$

Kuhn's theorem[15]

All three classes are equivalent in games of *perfect recall*.

$\hookrightarrow$ **Requires access to infinite memory and infinite support for distributions.**

[15] Aumann, "Mixed and Behavior Strategies in Infinite Extensive Games", 1964.

What about finite-memory strategies?

Mealy machine $\mathcal{M} = \{M, m_{\mathsf{init}}, \alpha_{\mathsf{nxt}}, \alpha_{\mathsf{up}}\}$:

- M is the set of memory states,
- m_{init} is the initial state,
- $\alpha_{\mathsf{nxt}} \colon M \times V \to V$ is the next-action function,
- $\alpha_{\mathsf{up}} \colon M \times V \to M$ is the update function.

What about finite-memory strategies?

Mealy machine $\mathcal{M} = \{M, m_{\mathsf{init}}, \alpha_{\mathsf{nxt}}, \alpha_{\mathsf{up}}\}$:

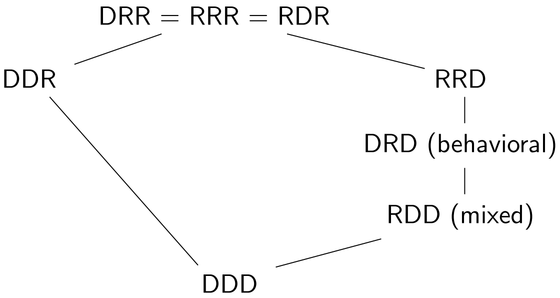
- M is the set of memory states,
- m_{init} is the initial state,
- $\alpha_{\mathsf{nxt}} \colon M \times V \to V$ is the next-action function,
- $\alpha_{\mathsf{up}} \colon M \times V \to M$ is the update function.

Stochastic Mealy machine $\mathcal{M} = \{M, \mu_{\mathsf{init}}, \alpha_{\mathsf{nxt}}, \alpha_{\mathsf{up}}\}$:

- M is the set of memory states,
- $\mu_{\mathsf{init}} \in \mathcal{D}(M)$ is the initial distribution,
- $\alpha_{\mathsf{nxt}} \colon M \times V \to \mathcal{D}(V)$ is the next-action function,
- $\alpha_{\mathsf{up}} \colon M \times V \to \mathcal{D}(M)$ is the update function.

$\Longrightarrow$ **Three ways to add randomness: initialization, outputs, and updates.**

Taxonomy[16] (1/2)



Classes XYZ where X, Y, Z $\in \{D, R\}$ where D stands for deterministic and R for random, and

- X characterizes the initialization,
- Y characterizes the next-action function,
- Z characterizes the update function.

[16] Main and Randour, "Different Strokes in Randomised Strategies: Revisiting Kuhn's Theorem Under Finite-Memory Assumptions", 2024.

Taxonomy (2/2)

This taxonomy holds from one-player deterministic games (no collapse) up to concurrent partial-information multi-player games (equivalences hold).

Taxonomy (2/2)

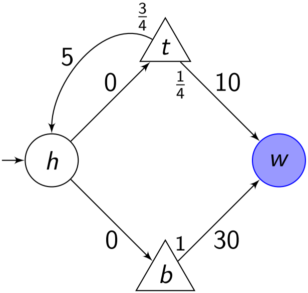
This taxonomy holds from one-player deterministic games (no collapse) up to concurrent partial-information multi-player games (equivalences hold).

$\hookrightarrow$ **Collapses may arise for restricted classes of objectives (WiP).**

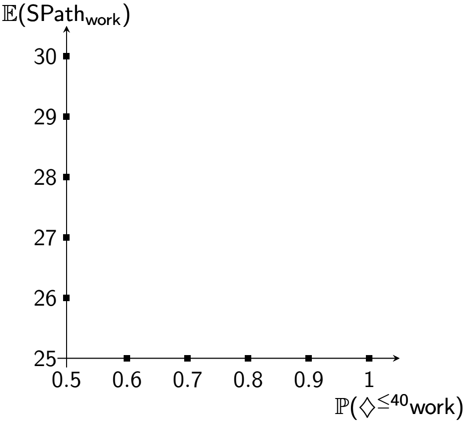
Multi-objectives MDPs (1/2)

We consider **two goals**:

- reaching work under 40 minutes with **high probability**;
- minimizing the **expectancy** of the time to reach work.



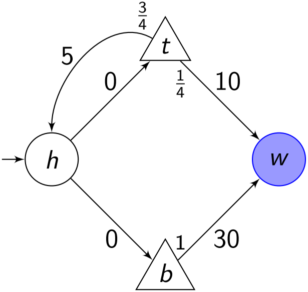
From *home*, take the *train* or *bike* to reach *work*.



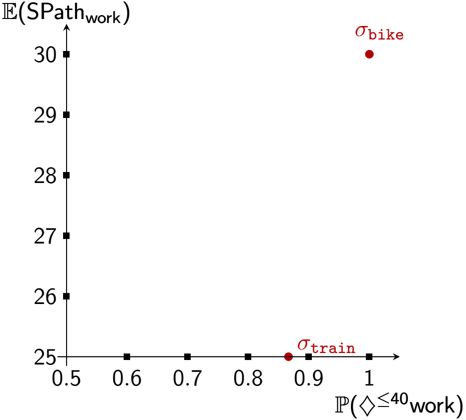
Multi-objectives MDPs (1/2)

We consider **two goals**:

- reaching work under 40 minutes with **high probability**;
- minimizing the **expectancy** of the time to reach work.



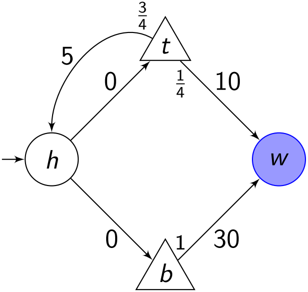
From *home*, take the *train* or *bike* to reach *work*.



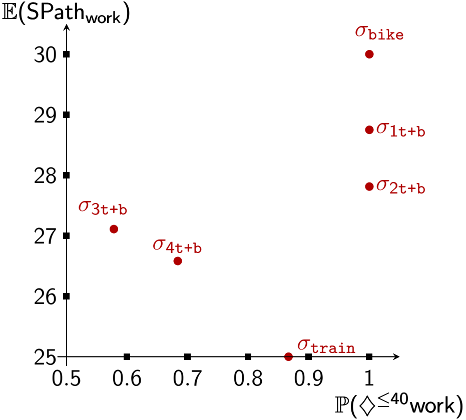
Multi-objectives MDPs (1/2)

We consider **two goals**:

- reaching work under 40 minutes with **high probability**;
- minimizing the **expectancy** of the time to reach work.



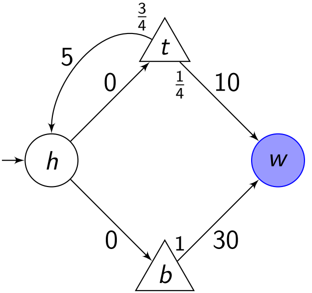
From *home*, take the *train* or *bike* to reach *work*.



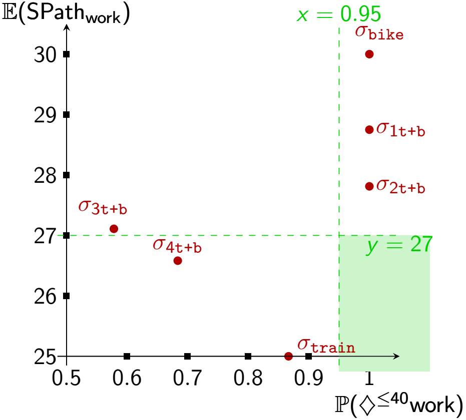
Multi-objectives MDPs (1/2)

We consider **two goals**:

- reaching work under 40 minutes with **high probability**;
- minimizing the **expectancy** of the time to reach work.



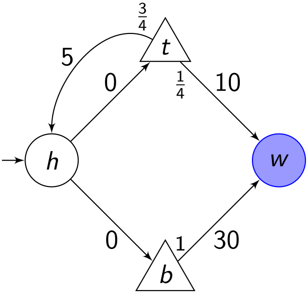
From *home*, take the *train* or *bike* to reach *work*.



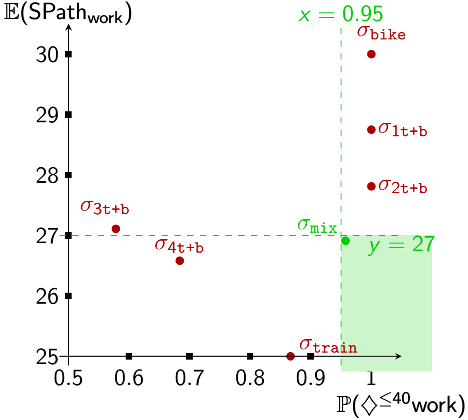
Multi-objectives MDPs (1/2)

We consider **two goals**:

- reaching work under 40 minutes with **high probability**;
- minimizing the **expectancy** of the time to reach work.



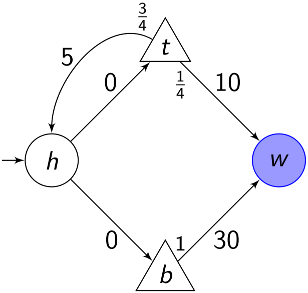
From *home*, take the *train* or *bike* to reach *work*.



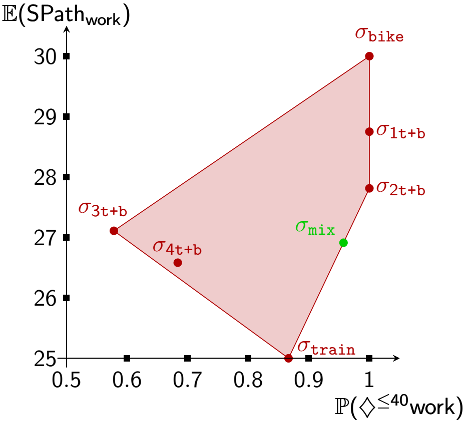
Multi-objectives MDPs (1/2)

We consider **two goals**:

- reaching work under 40 minutes with **high probability**;
- minimizing the **expectancy** of the time to reach work.



From *home*, take the *train* or *bike* to reach *work*.



Multi-objectives MDPs (2/2)

We are interested in the structure of this **payoff set**.

Multi-objectives MDPs (2/2)

We are interested in the structure of this **payoff set**.

Our result (WiP)

For *good* payoff functions ($\sim$ expectancy is well-defined),

1. the set of achievable payoffs coincide with the convex hull of *pure* payoffs;
2. we can approximate *any* strategy ε-closely by **mixing** a bounded number of *pure* strategies.

Multi-objectives MDPs (2/2)

We are interested in the structure of this **payoff set**.

Our result (WiP)

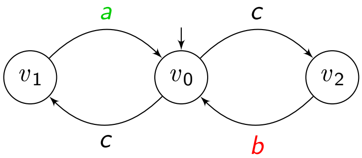
For *good* payoff functions ($\sim$ expectancy is well-defined),

1. the set of achievable payoffs coincide with the convex hull of *pure* payoffs;
2. we can approximate *any* strategy ε-closely by **mixing** a bounded number of *pure* strategies.

$\Longrightarrow$ **RDD-randomization is sufficient in most multi-objective MDPs.**

Trading memory for randomness

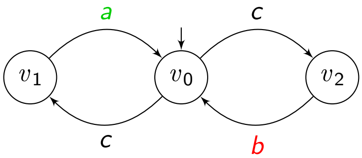
Recall this generalized Büchi game asking to see a and b infinitely often:



We need (a two-state) memory to win it with *pure* strategies.

Trading memory for randomness

Recall this generalized Büchi game asking to see *a* and *b* infinitely often:

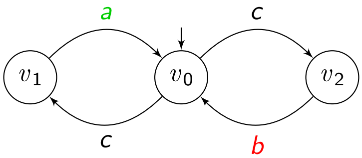


We need (a two-state) memory to win it with *pure* strategies.

But a (behavioral) randomized memoryless strategy suffices to win with probability one: playing v_1 and v_2 with non-zero probability ensures it.

Trading memory for randomness

Recall this generalized Büchi game asking to see *a* and *b* infinitely often:



We need (a two-state) memory to win it with *pure* strategies.

But a (behavioral) randomized memoryless strategy suffices to win with probability one: playing v_1 and v_2 with non-zero probability ensures it.

$\hookrightarrow$ **Memory can be traded for randomness for some classes of games/objectives.**[17]

[17] Chatterjee, de Alfaro, and Henzinger, "Trading Memory for Randomness", 2004; Chatterjee, Randour, and Raskin, "Strategy synthesis for multi-dimensional quantitative objectives", 2014.

An incomplete story

Leitmotiv

Simpler strategies are better (for controller synthesis).

An incomplete story

Leitmotiv

Simpler strategies are better (for controller synthesis).

But what is simple?

An incomplete story

Leitmotiv

Simpler strategies are better (for controller synthesis).

But what is simple?

Usual answer: small memory, no randomness.

An incomplete story

Leitmotiv

Simpler strategies are better (for controller synthesis).

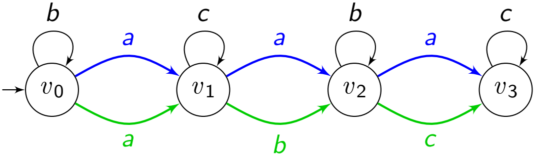
But what is simple?

Usual answer: small memory, no randomness.

$\hookrightarrow$ **Let us question that.**

All memoryless strategies are not created equal

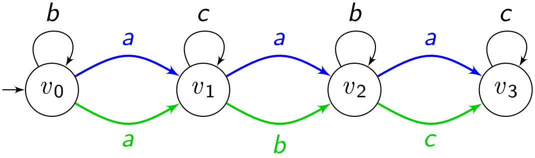
We want to reach v_3.



Intuitively, the blue strategy seems simpler than the green one.

All memoryless strategies are not created equal

We want to reach v_3.

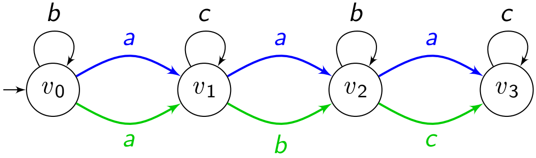


Intuitively, the blue strategy seems simpler than the green one.

▷ Yet both are represented as a trivial Mealy machine with a **single memory state**.

All memoryless strategies are not created equal

We want to reach v_3.

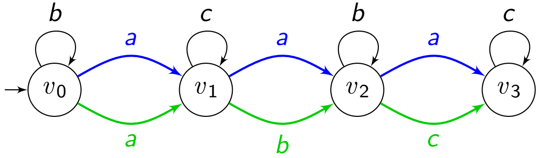


Intuitively, the blue strategy seems simpler than the green one.

- ▷ Yet both are represented as a trivial Mealy machine with a **single memory state**.
- ▷ The **representation of the next-action** function is mostly overlooked (basically a huge table).

All memoryless strategies are not created equal

We want to reach v_3.



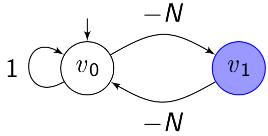
Intuitively, the blue strategy seems simpler than the green one.

- ▷ Yet both are represented as a trivial Mealy machine with a **single memory state**.
- ▷ The **representation of the next-action** function is mostly overlooked (basically a huge table).

$\hookrightarrow$ **Memoryless strategies can already be too large to represent in practice!**

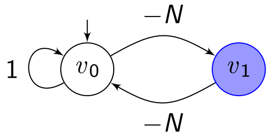
Memory is not always an issue

Multi-objectives games involving payoffs often require **exponential memory**. E.g., energy-Büchi objective with $N \in \mathbb{N}$.



Memory is not always an issue

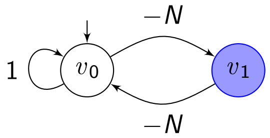
Multi-objectives games involving payoffs often require **exponential memory**. E.g., energy-Büchi objective with $N \in \mathbb{N}$.



▷ We need a pseudo-polynomial Mealy machine because it lacks structure.

Memory is not always an issue

Multi-objectives games involving payoffs often require **exponential memory**. E.g., energy-Büchi objective with $N \in \mathbb{N}$.

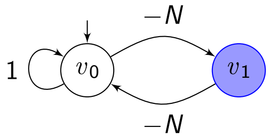


- ▷ We need a pseudo-polynomial Mealy machine because it lacks structure.

$\hookrightarrow$ **Polynomial representation if we allow the use of counters**.

Memory is not always an issue

Multi-objectives games involving payoffs often require **exponential memory**. E.g., energy-Büchi objective with $N \in \mathbb{N}$.



▷ We need a pseudo-polynomial Mealy machine because it lacks structure.

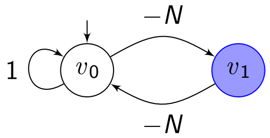
$\hookrightarrow$ **Polynomial representation if we allow the use of counters**.

Hot take

We should explore novel notions of **simplicity**, and consider *alternative representations* of strategies/controllers.

Memory is not always an issue

Multi-objectives games involving payoffs often require **exponential memory**. E.g., energy-Büchi objective with $N \in \mathbb{N}$.



- ▷ We need a pseudo-polynomial Mealy machine because it lacks structure.

$\hookrightarrow$ **Polynomial representation if we allow the use of counters**.

Hot take

We should explore novel notions of **simplicity**, and consider *alternative representations* of strategies/controllers.

$\hookrightarrow$ **We quickly survey a few ones in the next slides.**

Structurally-enriched Mealy machines

Idea:

- Augment Mealy machines with **data structures**: e.g., counters.[18]
- Avoid "flattening" structural information about the strategy: better understandability and closer to actual controllers.
- Link with James's talk about interval strategies in OC-MDPs.

$\Longrightarrow$ **Changes our way of thinking which strategies are complex or not.**

[18] Blahoudek et al., "Qualitative Controller Synthesis for Consumption Markov Decision Processes", 2020.

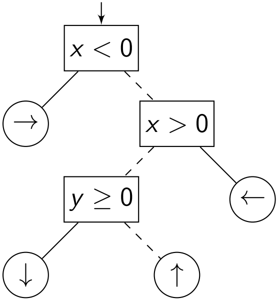
Decision trees

- Structured state-space (e.g., $\subset \mathbb{Z}^n$) and action-space.
- Learn a (possibly approximative) decision tree from a given **memoryless** strategy.
- More understandable and compact than huge action tables.
- More complex tests may reduce size but hinder readability.

Decision trees

- ▷ Structured state-space (e.g., $\subset \mathbb{Z}^n$) and action-space.
- ▷ Learn a (possibly approximative) decision tree from a given **memoryless** strategy.
- ▷ More understandable and compact than huge action tables.
- ▷ More complex tests may reduce size but hinder readability.

Toy example: trying to reach the center $(0, 0)$ of a 2D-grid.



instead of

x	y	action
0	1	$\downarrow$
0	2	$\downarrow$
...	...	$\downarrow$
-1	0	$\rightarrow$
-1	1	$\rightarrow$
...	...	...

Decision trees

- ▷ Structured state-space (e.g., $\subset \mathbb{Z}^n$) and action-space.
- ▷ Learn a (possibly approximative) decision tree from a given **memoryless** strategy.
- ▷ More understandable and compact than huge action tables.
- ▷ More complex tests may reduce size but hinder readability.

Works well in practice... [19]

...starting from a given memoryless strategy.

[19] Brazdil, Chatterjee, Chmelik, et al., "Counterexample Explanation by Learning Small Strategies in Markov Decision Processes", 2015; Brazdil, Chatterjee, Kretinsky, et al., "Strategy Representation by Decision Trees in Reactive Synthesis", 2018.

Other alternatives

- **Programmatic representations.**
  - ▷ Closer to realistic code, understandable.
  - ▷ Strongly linked to the input format of the problem (e.g., PRISM code[20]), hard to generalize.
- **Models inspired by Turing machines.**
  - ▷ Powerful but hard to work with.
  - ▷ Tentative notion of decision speed.[21]
- **Neural networks.**
  - ▷ Prevalent in RL.
  - ▷ Hard to understand and verify.
  - ▷ Can be coupled with finite-state-machine abstractions.[22]

[22] Shabadi, Fijalkow, and Matricon, "Theoretical foundations for programmatic reinforcement learning", 2024.

[22] Gelderie, "Strategy machines: representation and complexity of strategies in infinite games", 2014.

[22] Carr, Jansen, and Topcu, "Verifiable RNN-Based Policies for POMDPs Under Temporal Logic Constraints", 2020.

Wrap-up

Focus

Complexity of strategies in controller synthesis.

Wrap-up

Focus

Complexity of strategies in controller synthesis.

Mealy machines are a powerful tool from a theoretical standpoint.

- ▷ High-level picture w.r.t. **memory** and **randomness**.

Wrap-up

Focus

Complexity of strategies in controller synthesis.

Mealy machines are a powerful tool from a theoretical standpoint.

- ▷ High-level picture w.r.t. **memory** and **randomness**.

$\hookrightarrow$ **Many questions are still open!**

Wrap-up

Focus

Complexity of strategies in controller synthesis.

Mealy machines are a powerful tool from a theoretical standpoint.

- ▷ High-level picture w.r.t. **memory** and **randomness**.

$\hookrightarrow$ **Many questions are still open!**

Strategy complexity $\neq$ representation complexity.

Wrap-up

Focus

Complexity of strategies in controller synthesis.

Mealy machines are a powerful tool from a theoretical standpoint.

- ▷ High-level picture w.r.t. **memory** and **randomness**.

$\hookrightarrow$ **Many questions are still open!**

Strategy complexity $\neq$ representation complexity.

Take-home message

We need a **proper theory of complexity**, and a **toolbox of different representations**.

$\hookrightarrow$ **Ongoing project ControlleRS.**

Thank you! Any question?