Different Strokes in Randomised Strategies

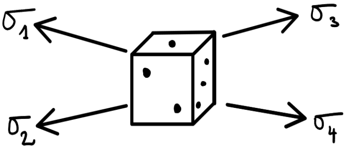
James C. A. Main Mickael Randour

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

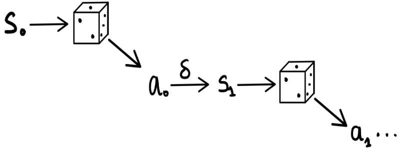
June 12, 2023

Talk overview

- We discuss games on graphs and randomised strategies.
- In general, such strategies can be defined in different ways.



Mixed strategies



Behavioural strategies

- In general, these two classes of strategies are not comparable.
- Kuhn's theorem [Aum64][1]: in games of perfect recall any mixed strategy has an equivalent behavioural strategy and vice-versa.

In this talk

We classify randomised finite-memory strategies and illustrate situations where more randomisation power is needed.

[1] Aumann, "28. Mixed and Behavior Strategies in Infinite Extensive Games".

Table of contents

Table of contents

Reactive systems

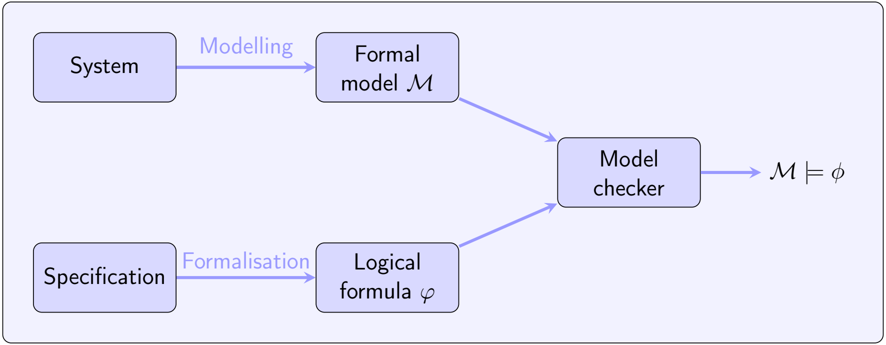
- Reactive systems are systems that interact continuously with an uncontrollable environment.

An automated vacuum cleaner is an example of a reactive system.

- Bugs in reactive systems can be notoriously hard to detect as it is not possible to test all possible sequences of inputs from the environment.

Model checking

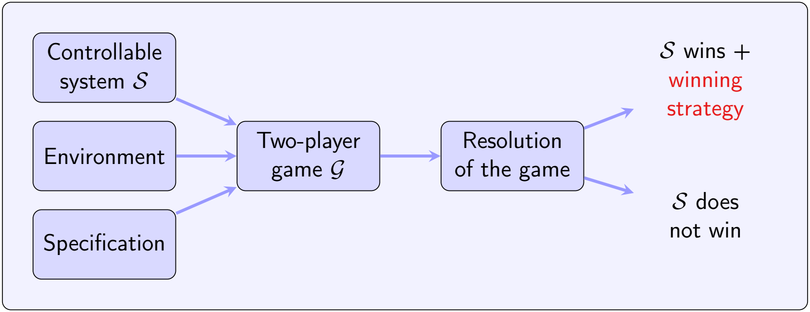
- Model checking provides mathematical guarantees on the behaviour of a system.



- Model checking requires a complete model of the system.

Reactive synthesis via game theory

- Reactive synthesis consists in the automatic synthesis of a controller for a system that ensures some specification.



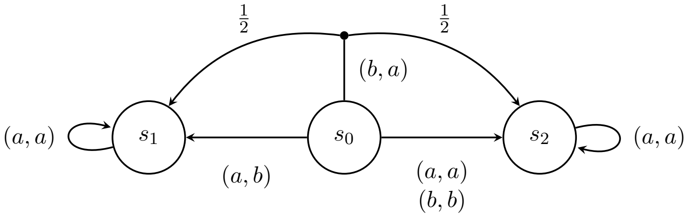
- Strategies in games are formal blueprints for controllers [Ran12][2].
- The classical representation is based on finite automata.

[2] Randour, "Automated Synthesis of Reliable and Efficient Systems Through Game Theory: A Case Study".

Table of contents

Concurrent games on finite graphs

We consider two-player stochastic concurrent games on finite graphs.



Essential characteristics

- Finite state space S and action spaces A_1 for $\mathcal{P}_1$, A_2 for $\mathcal{P}_2$.
- Partial probabilistic transition function $\delta \colon S \times A_1 \times A_2 \to \mathcal{D}(S)$.
- No deadlocks.
- A game is turn-based, if in all states, some player has only one action.

Plays, histories and objectives

- A play is a sequence $s_0 a_0^{(1)} a_0^{(2)} s_1 \ldots \in (SA_1A_2)^\omega$ obtained via the rules described previously.
- A history is a prefix of a play ending in a state.
- An objective for a player is defined as a set of plays, which describes the desired behaviour for the system. Classical objectives include:
  - reachability: the goal is reaching a set of targets state;
  - safety: the goal is to avoid a set of unsafe states.

Strategies

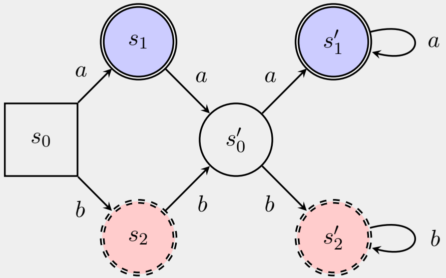
Definition

A (behavioural) strategy of $\mathcal{P}_i$ is a function $\sigma_i \colon \mathsf{Hist}(\mathcal{G}) \to \mathcal{D}(A_i)$.

- Strategies can use both memory and randomisation in general.

Memory is necessary in general

Assume $\mathcal{P}_1$ ($\bigcirc$) wants to force visits to both $\{s_1, s_1'\}$ and $\{s_2, s_2'\}$.



Strategies

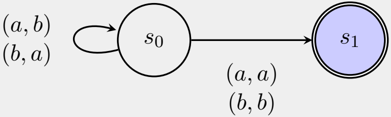
Definition

A (behavioural) strategy of $\mathcal{P}_i$ is a function $\sigma_i \colon \mathsf{Hist}(\mathcal{G}) \to \mathcal{D}(A_i)$.

- Strategies can use both memory and randomisation in general.

Randomisation is necessary in general

Assume $\mathcal{P}_1$ wants to visit $\{s_1\}$ almost-surely no matter the strategy of $\mathcal{P}_2$.



Outcomes of strategies and winning

- A play $s_0 a_0^{(1)} a_0^{(2)} \dots$ is called an outcome of a strategy σ_i of $\mathcal{P}_i$ if for all $k \in \mathbb{N}$, $\sigma_i(s_0 a_0^{(1)} a_0^{(2)} \dots s_k)(a_k^{(i)}) > 0$.
- Strategies σ_1 of $\mathcal{P}_1$ and σ_2 of $\mathcal{P}_2$ induce, from any initial state s_{init}, a probability distribution $\mathbb{P}^{\sigma_1,\sigma_2}_{s_{\mathsf{init}}}$ over plays.

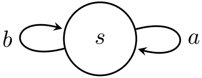
Winning in games

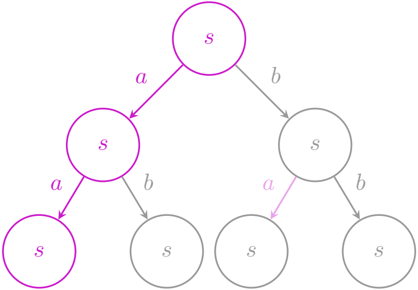
There exist different notions of winning. Given an objective Ω and an initial state s_{init}, we say that a strategy σ_1 of $\mathcal{P}_1$ is:

- surely winning if all of its outcomes starting in s_{init} are in Ω;
- almost-surely winning if for all strategies σ_2 of $\mathcal{P}_2$, $\mathbb{P}^{\sigma_1,\sigma_2}_{s_{\mathsf{init}}}(\Omega) = 1$;
- positively winning if for all strategies σ_2 of $\mathcal{P}_2$, $\mathbb{P}^{\sigma_1,\sigma_2}_{s_{\mathsf{init}}}(\Omega) > 0$.

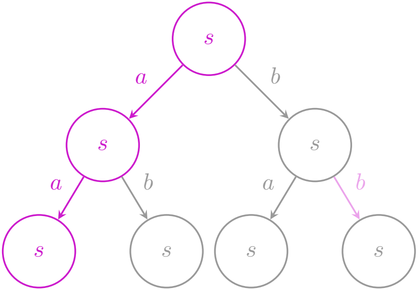
Comparing strategies

- Two different strategies of a player may exhibit the same behaviour.





Strategy that always uses action a surely.



Strategy that always uses action a and switches to action b if it occurs.

Outcome-equivalence

- When comparing two strategies, equality does not provide an accurate measure of equivalence.

Outcome-equivalence

Two strategies σ_1 and τ_1 of $\mathcal{P}_1$ are outcome-equivalent if for all strategies σ_2 of $\mathcal{P}_2$ and all initial states $s_{\mathsf{init}} \in S$, we have

$$\mathbb{P}_{s_{\mathsf{init}}}^{\sigma_1,\sigma_2} = \mathbb{P}_{s_{\mathsf{init}}}^{\tau_1,\sigma_2}.$$

- Equivalently, two strategies σ_1 and τ_1 are outcome-equivalent if, for all history h consistent with σ_1, $\sigma_1(h) = \tau_1(h)$.

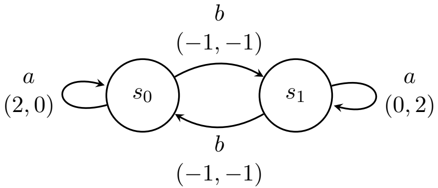
Table of contents

Finite-memory strategies

- In general, optimal strategies may require unlimited memory, which is unrealistic for practical applications.
- For instance, in the one-player game below, $\mathcal{P}_1$ cannot surely ensure a mean-payoff of $(1, 1)$, i.e., that

$$\liminf_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} (w^{(1)}(a_i), w^{(2)}(a_i)) \geq (1, 1)$$

using finite memory.



Randomised finite-memory strategies

- Finite-memory strategies are defined as finite automata with outputs.

Definition

A strategy σ_i of $\mathcal{P}_i$ is finite-memory if it can be induced by a stochastic Mealy machine $\mathcal{M} = (M, \mu_{\mathsf{init}}, \alpha_{\mathsf{next}}, \alpha_{\mathsf{up}})$ where

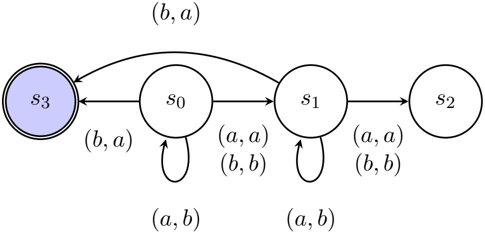
- M is a finite set of memory states;
- $\mu_{\mathsf{init}} \in \mathcal{D}(M)$ is an initial distribution;
- $\alpha_{\mathsf{next}} \colon M \times S \to \mathcal{D}(A)$ is a stochastic next-move function;
- $\alpha_{\mathsf{up}} \colon M \times S \times A_1 \times A_2 \to \mathcal{D}(M)$ is a stochastic memory update function.

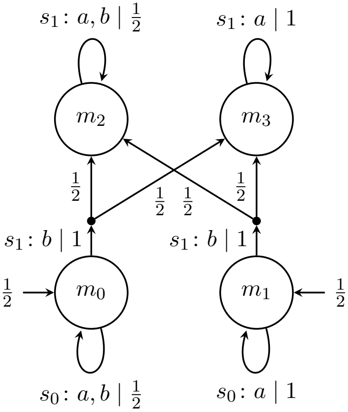
- We can classify Mealy machines following whether their initialisation, updates and outputs are randomised or deterministic.

Randomised finite-memory strategies

Example

- We illustrate a finite-memory strategy in the game below.



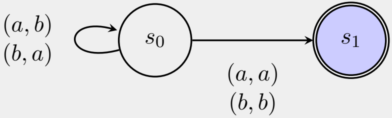


All classes of Mealy machines are not equally powerful

- Some classes of Mealy machines allow richer behaviours than others.

Example

In the game below, $\mathcal{P}_1$ cannot surely ensure that the state s_1 is visited almost-surely using finite-memory strategies derived from Mealy machines that use randomisation only in the initialisation.



A classification of finite-memory strategies

We use acronyms to define classes of Mealy machines: we use XYZ where X, Y, Z $\in$ {D, R} where D stands for deterministic and R for random, and

- X characterises initialisation,
- Y characterises outputs (next-move function),
- Z characterises updates.

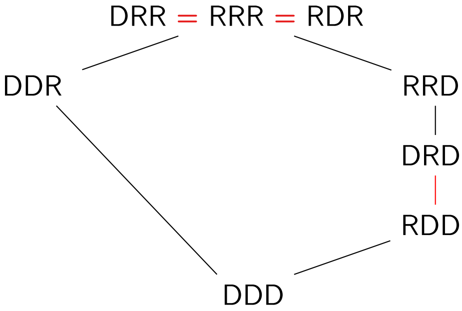
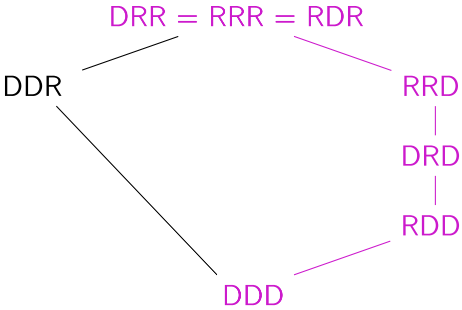


Table of contents

Distinguishing classes

- All non-inclusions can be witnessed in a one-player game with a single state and two actions.



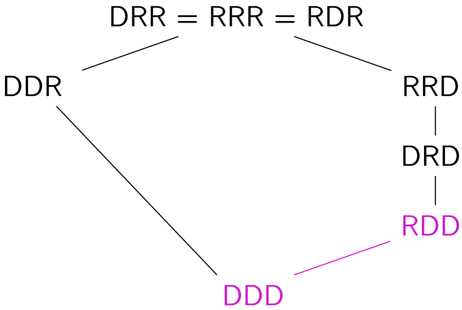
Goal of this section

Show the difference between classes by means of example objectives from the literature for which the larger class is sufficient and not the other.

Distinguishing classes

DDD vs. RDD

- We show that the classes DDD and RDD do not coincide.



Multi-objective reachability in Markov decision processes

DDD vs. RDD

- We consider one-player games with several reachability objectives Reach(F_1), ..., Reach(F_k) given by target sets $F_1, \dots, F_k$.
- A strategy σ_1 achieves at least $v \in [0,1]^k$ from an initial state s_{init} if $v_i \leq \mathbb{P}^{\sigma_1}_{s_{\text{init}}}(\text{Reach}(F_i))$ for all $1 \leq i \leq k$.
- RDD strategies can achieve vectors that DDD strategies cannot.

Example[3]

Let $F_1 = \{s_1\}$ and $F_2 = \{s_2\}$. The vector $(\frac{1}{2}, \frac{1}{2})$ cannot be achieved by a pure strategy, but can be achieved by an RDD strategy.

$$s_1 \xleftarrow{a} s_0 \xrightarrow{b} s_2$$

[3] Randour, Raskin, and Sankur, "Percentile queries in multi-dimensional Markov decision processes".

Multi-objective reachability in Markov decision processes

DDD vs. RDD

Theorem (Consequence of [EKVY08][4])

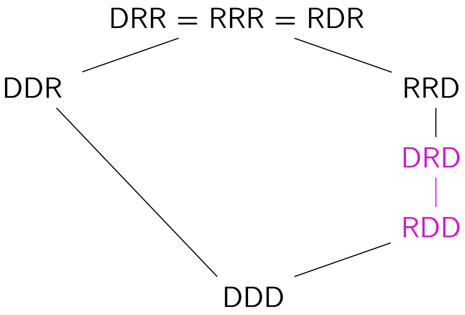
RDD strategies suffice to achieve any vector for multi-objective reachability with absorbing targets in Markov decision processes.

- The set of vectors that can be achieved by some strategy is a convex polyhedral set.
- The vertices of this set of vectors can be achieved by pure memoryless strategies.
- Any vector can be achieved by an RDD strategy that is randomly initialised to these memoryless strategies.

[4] Etessami et al., "Multi-Objective Model Checking of Markov Decision Processes".

Distinguishing classes
RDD vs. DRD

- We have seen previously that the classes RDD and DRD do not coincide.

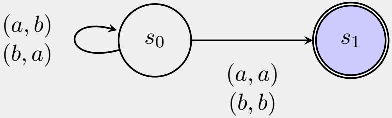


Concurrent reachability games

- In concurrent reachability games, RDD strategies may not suffice.

Example

There is no almost-surely winning RDD strategy for $\mathcal{P}_1$ for the reachability objective with target $\{s_1\}$.



- However, DRD strategies suffice to win almost-surely.

Theorem ([dAHK07][5])

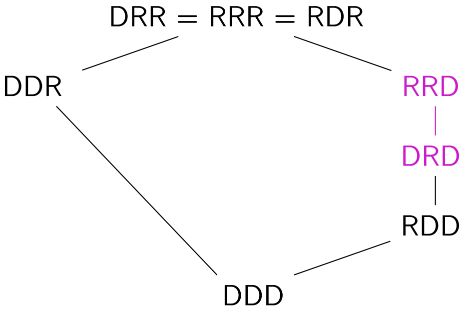
Memoryless randomised strategies (DRD strategies with one memory state) suffice to win almost-surely in concurrent reachability games.

[5] de Alfaro, Henzinger, and Kupferman, "Concurrent reachability games".

Distinguishing classes

DRD vs. RRD

- We show that the classes DRD and RRD do not coincide.

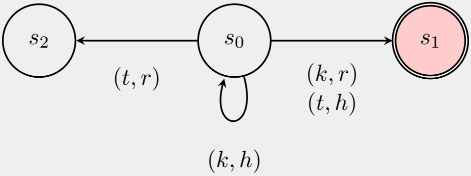


Concurrent safety games

- DRD strategies do not suffice to win positively in concurrent safety games.[6]

Example

There is no positively winning DRD strategy for $\mathcal{P}_1$ for the safety objective with bad state s_1.

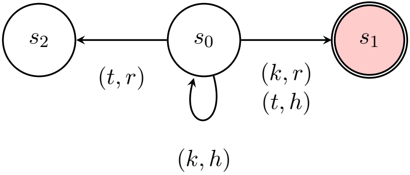


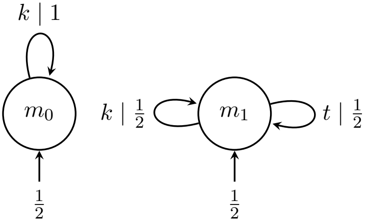
- However, there exists a positively winning RRD strategy.

[6] de Alfaro, Henzinger, and Kupferman, "Concurrent reachability games".

Concurrent safety games

- A positively winning strategy for the safety objective defined from s_1 is illustrated below.
- We only depict outputs and updates in s_0.





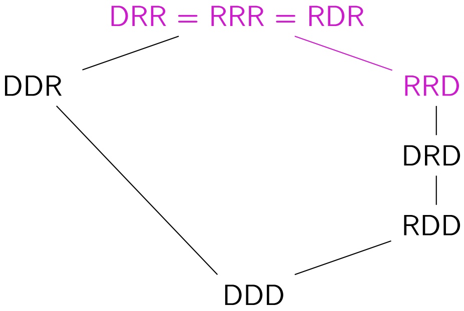
Theorem ([CDH10][7])

RRR strategies suffice to win positively in concurrent safety games.

[7] Cristau, David, and Horn, "How do we remember the past in randomised strategies?"

Distinguishing classes
RRD vs. RRR

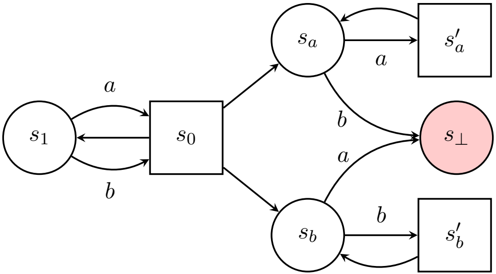
- We show that the classes RRD and RRR do not coincide.



- For this section, we assume that one of the players has imperfect information.

Safety games of imperfect information

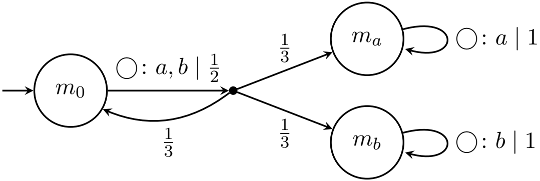
- We consider the safety objective to avoid visiting $s_\perp$.
- $\mathcal{P}_1$ can only observe his own actions and when it is their turn to play.
- We omit the actions of $\mathcal{P}_2$ to lighten the illustration.



- To win positively, $\mathcal{P}_1$ must have a positive probability of using a same action without ever changing again from any point on.

Safety games of imperfect information

- No RRD strategy has the property needed to win positively.
- The strategy below is positively winning for $\mathcal{P}_1$ in the previous game.



Theorem ([BGG17][8])

RRR strategies suffice to win positively in safety games of imperfect information.

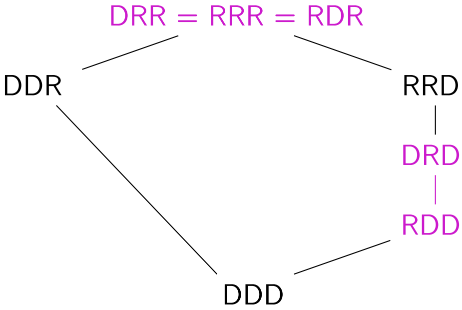
[8] Bertrand, Genest, and Gimbert, "Qualitative Determinacy and Decidability of Stochastic Games with Signals".

Table of contents

Describing inclusions

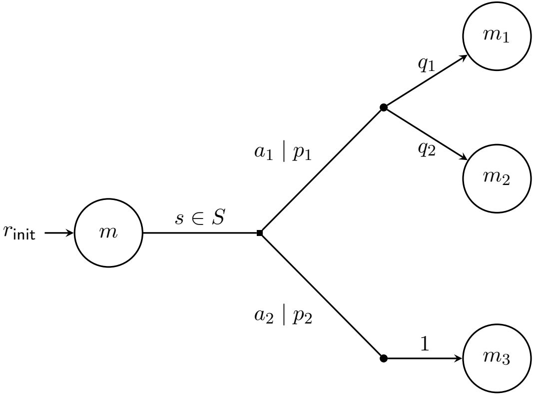
Goal of this section: non-trivial inclusions of the lattice

- RDD $\subseteq$ DRD,
- RRR $\subseteq$ DRR,
- RRR $\subseteq$ RDR.



Illustrating a finite-memory strategy

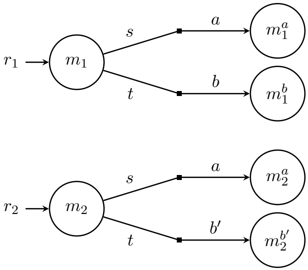
- In the sequel, we will illustrate fragments of Mealy machines for $\mathcal{P}_i$ as follows.
- For the sake of readability, we assume that memory updates do not depend on actions of $\mathcal{P}_{3-i}$.

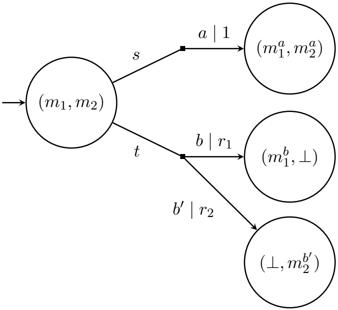


RDD $\subseteq$ DRD: trading random initialisation for outputs

We fix an RDD Mealy machine $\mathcal{M} = (M, \mu_{\mathsf{init}}, \alpha_{\mathsf{next}}, \alpha_{\mathsf{up}})$.

- We use an adaptation of the subset construction to go from $\mathcal{M}$ to a DRD Mealy machine.
- State space of functions $f\colon \mathsf{supp}(\mu_{\mathsf{init}}) \to (M \cup \{\perp\})$:
  - We simulate the strategy from each initial state.
  - If an action is inconsistent with one of the simulations, we stop it (symbolised by $\perp$).

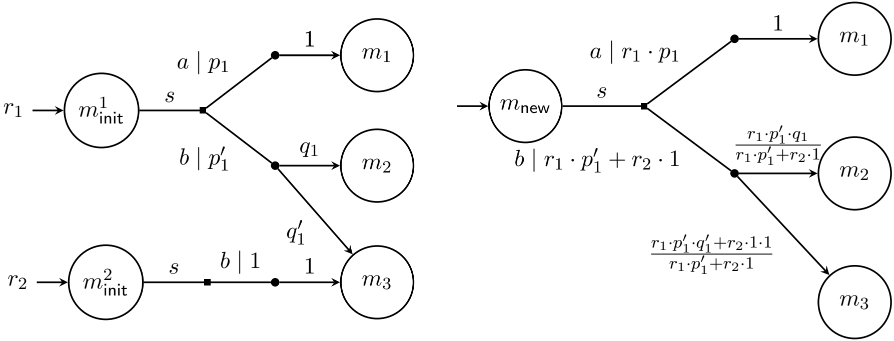




RRR ⊆ DRR: determinising initialisation

We fix an RRR Mealy machine $\mathcal{M} = (M, \mu_{\mathsf{init}}, \alpha_{\mathsf{next}}, \alpha_{\mathsf{up}})$.

- To derive a DRR Mealy machine from $\mathcal{M}$, we add a new initial state m_{new} to the memory state space.
- We use stochastic updates to return to $\mathcal{M}$ from m_{new}. Transition probabilities are chosen so the distribution over memory states is the same in $\mathcal{M}$ and the DRR Mealy machine after the first step.



RRR $\subseteq$ RDR: determinising outputs

We fix an RRR Mealy machine $\mathcal{M} = (M, \mu_{\mathsf{init}}, \alpha_{\mathsf{next}}, \alpha_{\mathsf{up}})$.

- To derive an RDR Mealy machine from $\mathcal{M}$, we expand the state space by augmenting memory states with pure memoryless strategies $\sigma_i \colon S_i \to A$.
- We use stochastic initialisation and updates to integrate the randomisation over actions in the transitions.

Naive construction $\rightsquigarrow$ memory state space grows by a factor of $|A|^{|S_i|}$

$\hookrightarrow$ We can do better:

Theorem

There exists an RDR Mealy machine with $|M| \cdot |S_i| \cdot |A|$ states whose induced strategy is outcome-equivalent to $\mathcal{M}$.

RRR $\subseteq$ RDR: choosing pure memoryless strategies

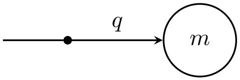
- Consider a game such that $S_i = \{s_1, s_2, s_3\}$, and $A = \{a_1, a_2, a_3\}$. Assume that for a memory state $m \in M$, we have:
  - $\alpha_{\mathsf{next}}(m, s_1)(a_1) = \alpha_{\mathsf{next}}(m, s_1)(a_2) = \frac{1}{2}$;
  - $\alpha_{\mathsf{next}}(m, s_2)(a_1) = \alpha_{\mathsf{next}}(m, s_2)(a_2) = \alpha_{\mathsf{next}}(m, s_2)(a_3) = \frac{1}{3}$;
  - $\alpha_{\mathsf{next}}(m, s_3)(a_1) = \frac{1}{3}$, $\alpha_{\mathsf{next}}(m, s_3)(a_2) = \frac{1}{6}$ and $\alpha_{\mathsf{next}}(m, s_3)(a_3) = \frac{1}{2}$.
- We represent the actions in a table to derive the pure memoryless strategies and their probabilities.

<table>
<tr><td>s_1</td><td colspan="2">a_1</td><td colspan="2">a_2</td></tr>
<tr><td>s_2</td><td>a_1</td><td colspan="2">a_2</td><td>a_3</td></tr>
<tr><td>s_3</td><td>a_1</td><td>a_2</td><td colspan="2">a_3</td></tr>
<tr><td>σ_k</td><td>σ_1</td><td>σ_2</td><td>σ_3</td><td>σ_4</td></tr>
</table>

$x_1 = 0$ $x_2 = \frac{1}{3}$ $x_3 = \frac{1}{2}$ $x_4 = \frac{2}{3}$ $x_5 = 1$

RRR $\subseteq$ RDR: exploiting the memoryless strategies

- For each memory state $m \in M$, we determine pure memoryless strategies $\sigma_1^m, \ldots, \sigma_{\ell(m)}^m$ and their respective probabilities $p_1^m, \ldots, p_{\ell(m)}^m$.
- We split transitions that enter m into transitions that go to the states (m, σ_j^m): a transition of probability q into m yields a transition with probability $q \cdot p_j^m$ into (m, σ_j^m).



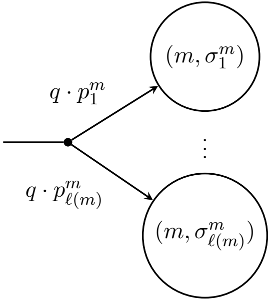


Table of contents

Taxonomy in broader settings

- We have only considered two-player games.
- However, the classification we have discussed here applies also in multi-player games.
- It also applies in games of imperfect information assuming a player can see their own actions.
  - It is not necessary to see the states themselves.
  - For the inclusion RDD $\subseteq$ DRD, we rely on the visibility of actions in our subset construction.
  - For the inclusion RRR $\subseteq$ DRR, we also use the visibility of actions in conditional probabilities.
- However, if actions cannot be observed, then the two inclusions mentioned above do not hold.

Downsides of more powerful strategies

- RRR strategies can induce strategies that are complicated to understand in general.
  - This is undesirable in contexts where explainability of the behaviour of strategies is important.
- RRR strategies are less amenable to computational analyses.
  - Determining, given an RRR strategy of $\mathcal{P}_1$, an initial state and a set of states F, whether the strategy is positively winning for Safe(F) is undecidable[9], even in turn-based games.
  - Therefore, it is hard to verify a given RRR strategy.

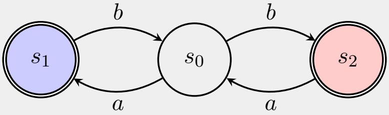
[9] Gimbert and Oualhadj, "Probabilistic Automata on Finite Words: Decidable and Undecidable Problems".

Advantages of more powerful strategies

- Allowing more randomisation allows one to capture more interesting behaviours.
- In some cases, memory can be traded off with randomisation; choosing a richer model of randomised finite-memory strategies yields more concise strategies[10].

Example

$\mathcal{P}_1$ wants to visit the states s_1 and s_2 infinitely often almost-surely.



Memory is necessary to play without randomisation but not otherwise.

[10] Chatterjee, de Alfaro, and Henzinger, "Trading Memory for Randomness"; Horn, "Random Fruits on the Zielonka Tree".

References I

Aumann, Robert J . "28. Mixed and Behavior Strategies in Infinite Extensive Games". In: *Advances in Game Theory. (AM-52), Volume 52*. Princeton University Press, 2016, pp. 627–650. DOI: doi:10.1515/9781400882014-029. URL: https://doi.org/10.1515/9781400882014-029.

Bertrand, Nathalie, Blaise Genest, and Hugo Gimbert. "Qualitative Determinacy and Decidability of Stochastic Games with Signals". In: *J. ACM* 64.5 (2017), 33:1–33:48. DOI: 10.1145/3107926. URL: https://doi.org/10.1145/3107926.

Chatterjee, Krishnendu, Luca de Alfaro, and Thomas A. Henzinger. "Trading Memory for Randomness". In: *1st International Conference on Quantitative Evaluation of Systems (QEST 2004), 27-30 September 2004, Enschede, The Netherlands*. IEEE Computer Society, 2004, pp. 206–217. DOI: 10.1109/QEST.2004.1348035. URL: https://doi.org/10.1109/QEST.2004.1348035.

References II

Cristau, Julien, Claire David, and Florian Horn. "How do we remember the past in randomised strategies?" In: *Proceedings First Symposium on Games, Automata, Logic, and Formal Verification, GANDALF 2010, Minori (Amalfi Coast), Italy, 17-18th June 2010*. Ed. by Angelo Montanari, Margherita Napoli, and Mimmo Parente. Vol. 25. EPTCS. 2010, pp. 30–39. DOI: 10.4204/EPTCS.25.7. URL: https://doi.org/10.4204/EPTCS.25.7.

de Alfaro, Luca, Thomas A. Henzinger, and Orna Kupferman. "Concurrent reachability games". In: *Theor. Comput. Sci.* 386.3 (2007), pp. 188–217. DOI: 10.1016/j.tcs.2007.07.008. URL: https://doi.org/10.1016/j.tcs.2007.07.008.

Etessami, Kousha et al. "Multi-Objective Model Checking of Markov Decision Processes". In: *Log. Methods Comput. Sci.* 4.4 (2008). DOI: 10.2168/LMCS-4(4:8)2008.

References III

Gimbert, Hugo and Youssouf Oualhadj. "Probabilistic Automata on Finite Words: Decidable and Undecidable Problems". In: *Automata, Languages and Programming, 37th International Colloquium, ICALP 2010, Bordeaux, France, July 6-10, 2010, Proceedings, Part II*. Ed. by Samson Abramsky et al. Vol. 6199. Lecture Notes in Computer Science. Springer, 2010, pp. 527–538. DOI: 10.1007/978-3-642-14162-1_44. URL: https://doi.org/10.1007/978-3-642-14162-1_44.

Horn, Florian. "Random Fruits on the Zielonka Tree". In: *26th International Symposium on Theoretical Aspects of Computer Science, STACS 2009, February 26-28, 2009, Freiburg, Germany, Proceedings*. Ed. by Susanne Albers and Jean-Yves Marion. Vol. 3. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Germany, 2009, pp. 541–552. DOI: 10.4230/LIPIcs.STACS.2009.1848. URL: https://doi.org/10.4230/LIPIcs.STACS.2009.1848.

References IV

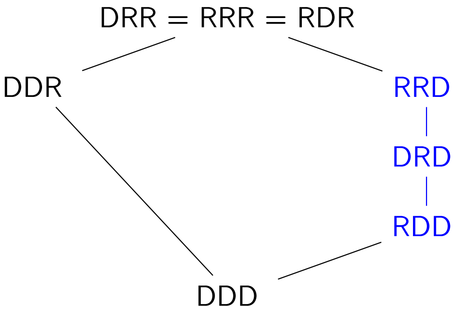
Randour, Mickael. "Automated Synthesis of Reliable and Efficient Systems Through Game Theory: A Case Study". English. In: *Proc. of ECCS 2012*. Springer Proceedings in Complexity XVII. Springer, 2013, pp. 731–738. ISBN: 978-3-319-00394-8. DOI: 10.1007/978-3-319-00395-5_90.

Randour, Mickael, Jean-François Raskin, and Ocan Sankur. "Percentile queries in multi-dimensional Markov decision processes". In: *Formal Methods Syst. Des.* 50.2-3 (2017), pp. 207–248. DOI: 10.1007/s10703-016-0262-7. URL: https://doi.org/10.1007/s10703-016-0262-7.

Differences between classes

We illustrate the strictness properties in a one-player game with a single state and two actions.

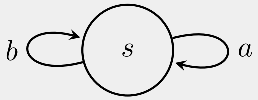
- The chain of inclusions DDD $\subsetneq$ RDD $\subsetneq$ DRD $\subsetneq$ RRD $\subsetneq$ RRR is strict.
- It holds that DDR $\not\subseteq$ RRD and RDD $\not\subseteq$ DDR.



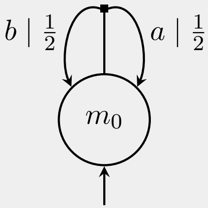
Strictness: $\mathrm{RDD} \subsetneq \mathrm{DRD}$

- In a one-player deterministic game, RDD strategies have finitely many outcomes.
- The DRD strategy depicted below has no RDD equivalent.

Game



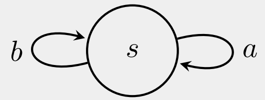
Witness of $\mathrm{RDD} \subsetneq \mathrm{DRD}$



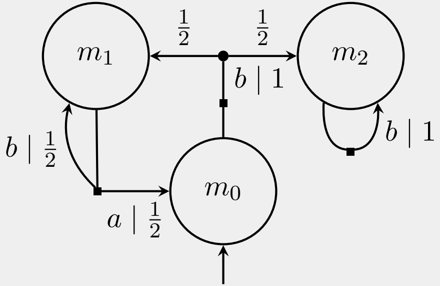
Strictness: $\mathrm{DDR} \not\subseteq \mathrm{RRD}$

- The number of memory states in which we can find ourselves as a play goes on cannot increase for an RRD strategy.
- To have a positive probability of never using a, we must eventually be in a memory state m such that $\alpha_{\mathsf{next}}(m, s)(a) = 0$ with positive probability.

Game



Can be adapted to witness $\mathrm{DDR} \not\subseteq \mathrm{RRD}$

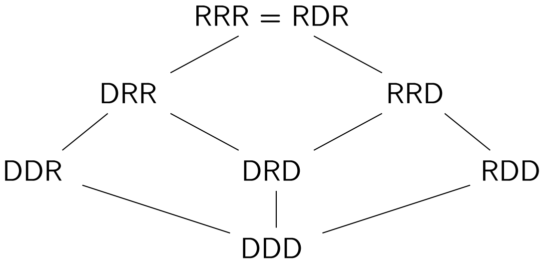


Collapses – invisible actions

What happens to the lattice in full generality ? If we assume nothing on the visibility of actions ?

- Two inclusions of our lattice no longer hold. We have:
  - RDD$\not\subseteq$DRD;
  - RRR$\not\subseteq$DRR (we even have RDD$\not\subseteq$DRR).
- Intuitively, for a strategy with deterministic outputs (i.e., in a subclass of RDR), the output actions are encoded in the Mealy machine itself.
  $\rightsquigarrow$ such strategies allow the same behaviours whether actions are visible or not.

General lattice: no hypotheses on actions



Subgame perfect equilibria and Kuhn's theorem

- In the statement of Kuhn's theorem and our classification, the output of the strategies along inconsistent branches histories are completely disregarded.
- In other words, our classification approach is not relevant for the study of subgame perfect equilibria, for which these inconsistent histories are nonetheless taken in account.
- However, the output of a finite-memory strategy along an inconsistent history is not well-defined.