Different Strokes in Randomised Strategies: Revisiting Kuhn's Theorem Under Finite-memory Assumptions

James C. A. Main   Mickael Randour

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

Seminar – 31 May 2022

Introduction

- In general, one can define randomised strategies in different ways.
  - Mixed strategies randomise between pure strategies at the start.
  - Behavioural strategies randomly select an action at each step.
- In general, these two classes of strategies are not comparable.

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

Introduction

- In general, one can define randomised strategies in different ways.
  - Mixed strategies randomise between pure strategies at the start.
  - Behavioural strategies randomly select an action at each step.
- 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.

- There exist different definitions of randomised finite-memory strategies.
- However, they are not all equivalent.

Our contribution

An adaptation of Kuhn's theorem for finite-memory strategies.

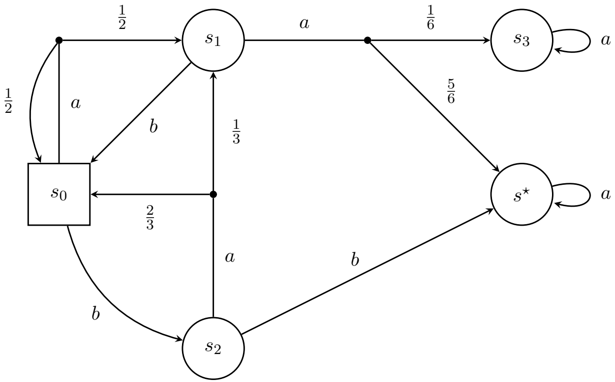
[1] Aumann, "28. Mixed and Behavior Strategies in Infinite Extensive Games".

Table of contents

Table of contents

Stochastic games of perfect information

We consider two-player stochastic games of perfect information.



Definitions

Definition

A stochastic game of perfect information is a tuple $\mathcal{G} = (S_1, S_2, A, \delta)$ where

- $S = S_1 \uplus S_2$ is a finite set of states, S_i is the set of $\mathcal{P}_i$ states;
- A is a finite set of actions;
- $\delta \colon S \times A \to \mathcal{D}(S)$ is a partial transition relation.

For all $s \in S$, let $A(s) = \{a \in A \mid \delta(s, a) \text{ is defined}\}$ denote the set of actions enabled in s. We assume that in each state $s \in S$, there is at least one enabled action.

- Play: sequence $s_0 a_0 s_1 \ldots$ where for all $k \in \mathbb{N}$, $a_k \in A(s_k)$ and $\delta(s_k, a_k)(s_{k+1}) > 0$.
- History: prefix of a play ending in a state. We write $\mathsf{Hist}_i(\mathcal{G})$ for the set of histories ending in S_i.

Strategies

Definition

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

Given two strategies σ_1 and σ_2, and an initial state $s_{\mathsf{init}} \in S$, we define a probability distribution on the set of plays in the usual way: for any history $h = s_0 a_0 s_1 \dots s_n$ with $s_0 = s_{\mathsf{init}}$, we set

$$\mathbb{P}_s^{\sigma_1,\sigma_2}(\mathsf{Cyl}(h)) = \prod_{k=0}^{n-1} \sigma_{i(k)}(s_0 a_0 \dots s_k) \cdot \delta(s_k, a_k, s_{k+1})$$

where $\mathsf{Cyl}(h)$ is the set of plays with h as a prefix, and $i(k) = 1$ if $s_k \in S_1$ and 2 otherwise.

How to compare strategies ?

- We compare strategies independently of any objective or payoff.
- Equality is too restrictive: two different strategies may induce the same behaviour in practice.

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 $s_{\text{init}} \in S$, we have

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

- Outcome-equivalence of strategies preserves optimality of strategies.

Table of contents

Finite-memory strategies

- In general, strategies can use unlimited memory.
  $\rightsquigarrow$ unrealistic in practice
- We consider finite-memory strategies.

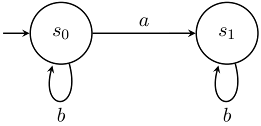
Definition

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

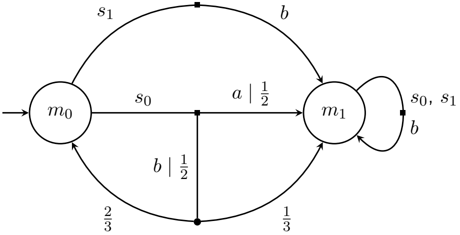
- M is a finite set of memory states;
- $\mu_{\mathsf{init}} \in \mathcal{D}(M)$ is an initial distribution;
- $\alpha_{\mathsf{up}} \colon M \times S \times A \to \mathcal{D}(M)$ is a stochastic memory update function;
- $\alpha_{\mathsf{next}} \colon M \times S_i \to \mathcal{D}(A)$ is a stochastic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

Playing with Mealy machines

- Consider the following game.



- An example of a Mealy machine encoding a $\mathcal{P}_1$ strategy in this game is given hereunder.



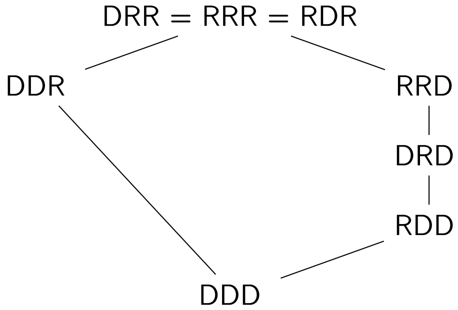
Classifying finite-memory strategies

- Given a Mealy machine $\mathcal{M} = (M, \mu_{\mathsf{init}}, \alpha_{\mathsf{up}}, \alpha_{\mathsf{next}})$, we can formally define the strategy it induces.
- We can classify finite-memory strategies depending on the form of Mealy machines that induce them.
- Depending on whether the initialisation, outputs or updates of Mealy machines are deterministic or randomised, the expressive power of the matching class of strategies varies.

Classifying 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.



Illustrating a finite-memory strategy

In the sequel, we will illustrate fragments of Mealy machines for $\mathcal{P}_i$ as follows.

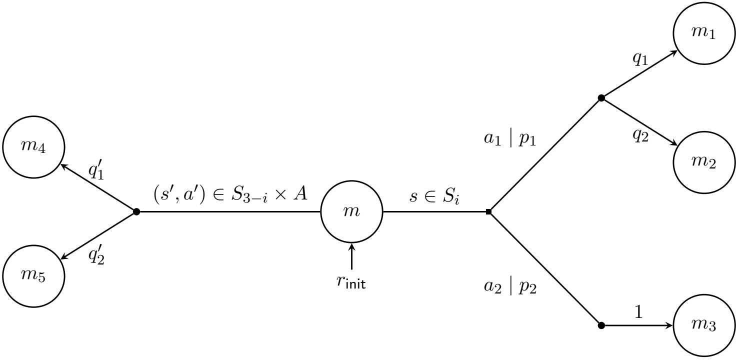
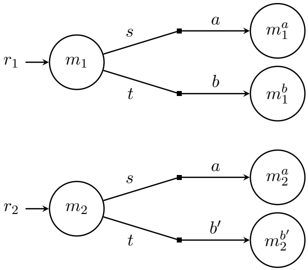


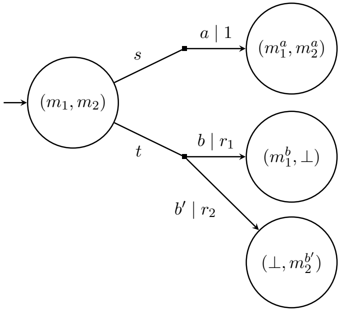
Table of contents

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

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

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

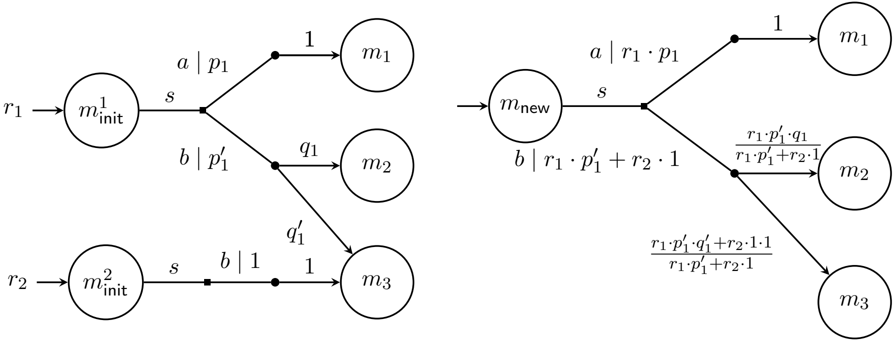




RRR $\subseteq$ DRR: determinising initialisation

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

- 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 $\mathcal{M} = (M, \mu_{\text{init}}, \alpha_{\text{up}}, \alpha_{\text{next}})$.

- To derive a 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 $\alpha_{\mathsf{next}}(m, s_3)(a_3) = \frac{1}{2}$.

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:
  - $\alpha_{\text{next}}(m, s_1)(a_1) = \alpha_{\text{next}}(m, s_1)(a_2) = \frac{1}{2}$;
  - $\alpha_{\text{next}}(m, s_2)(a_1) = \alpha_{\text{next}}(m, s_2)(a_2) = \alpha_{\text{next}}(m, s_2)(a_3) = \frac{1}{3}$;
  - $\alpha_{\text{next}}(m, s_3)(a_1) = \frac{1}{3}$, $\alpha_{\text{next}}(m, s_3)(a_2) = \frac{1}{6}$ and $\alpha_{\text{next}}(m, s_3)(a_3) = \frac{1}{2}$.
- We represent the actions in a table to derive the pure memoryless strategies and their probabilities.

<table>
<tr><td>s_1</td><td colspan="3">a_1</td><td colspan="3">a_2</td></tr>
<tr><td>s_2</td><td colspan="2">a_1</td><td colspan="2">a_2</td><td colspan="2">a_3</td></tr>
<tr><td>s_3</td><td colspan="2">a_1</td><td>a_2</td><td colspan="3">a_3</td></tr>
</table>

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:
  - $\alpha_{\text{next}}(m, s_1)(a_1) = \alpha_{\text{next}}(m, s_1)(a_2) = \frac{1}{2}$;
  - $\alpha_{\text{next}}(m, s_2)(a_1) = \alpha_{\text{next}}(m, s_2)(a_2) = \alpha_{\text{next}}(m, s_2)(a_3) = \frac{1}{3}$;
  - $\alpha_{\text{next}}(m, s_3)(a_1) = \frac{1}{3}$, $\alpha_{\text{next}}(m, s_3)(a_2) = \frac{1}{6}$ and $\alpha_{\text{next}}(m, s_3)(a_3) = \frac{1}{2}$.
- We represent the actions in a table to derive the pure memoryless strategies and their probabilities.

<table>
<tr><td>s_1</td><td colspan="3">a_1</td><td colspan="3">a_2</td></tr>
<tr><td>s_2</td><td colspan="2">a_1</td><td colspan="2">a_2</td><td colspan="2">a_3</td></tr>
<tr><td>s_3</td><td colspan="2">a_1</td><td>a_2</td><td colspan="3">a_3</td></tr>
</table>

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>

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

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).

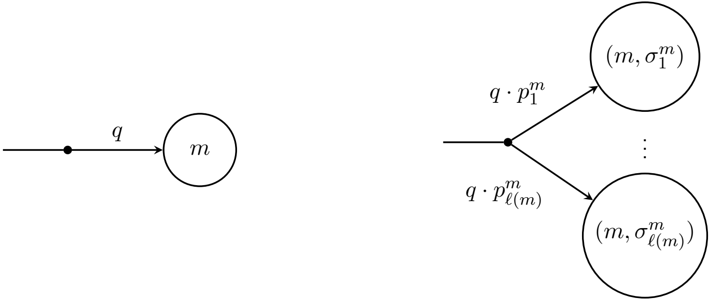
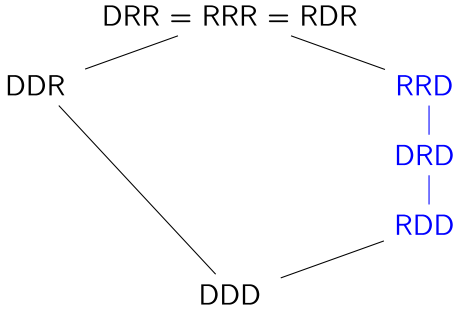


Table of contents

Differences between classes

We discuss the following aspects:

- 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.



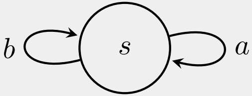
Strictness: RDD $\subsetneq$ DRD

In one-player deterministic games:

- there are finitely many outcomes for any $\mathcal{P}_i$ RDD strategy;
- there may be infinitely many outcomes for a $\mathcal{P}_i$ DRD strategy.

Example

The memoryless strategy $\sigma_1 \colon \{s\} \to \mathcal{D}(\{a, b\})$ such that $\sigma_1(s)$ is the uniform distribution over $\{a, b\}$ can be induced by a DRD Mealy machine, but not by a RDD one.



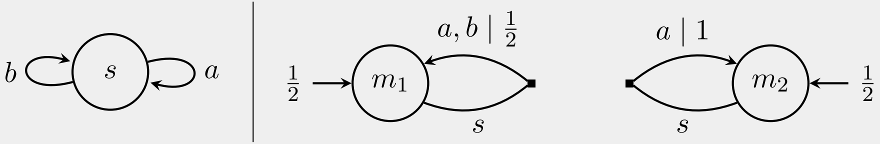
Strictness: DRD $\subsetneq$ RRD

In one-player deterministic games:

- an RRD strategy can be designed to ensure some action is taken at each step with positive probability but has a positive probability of never being taken;
- a DRD strategy that attempts a certain action with a positive probability at each step will almost surely play it.

Example

The following RRD Mealy machine has no equivalent DRD machine.



Strictness: RRD $\subsetneq$ RRR

In two-player deterministic games and Markov decision processes:

- stochastic updates allow RRR Mealy machines to induce strategies that suggest actions with probabilities arbitrarily close to zero after histories controlled by $\mathcal{P}_{3-i}$;

A DDR example

In memory state m_c, $\mathcal{P}_i$ plays action c and does not change memory states $\rightsquigarrow$ no matching RRD Mealy machine.

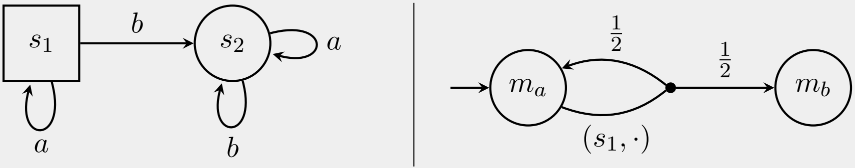


Table of contents

Taxonomy in settings of partial information

- Up to now, we have discussed a classification of strategies in a setting of perfect information.

$\rightsquigarrow$ Can we weaken this hypothesis ?

- 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.

Partial information

The classification holds in games where $\mathcal{P}_i$ can see their actions and distinguish the owner of states from their observations.

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.

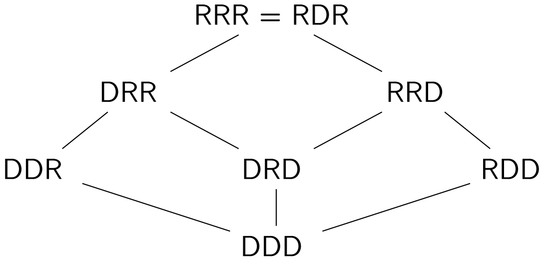
Main, James C. A. and Mickael Randour. "Different strokes in randomised strategies: Revisiting Kuhn's theorem under finite-memory assumptions". In: *CoRR* abs/2201.10825 (2022). arXiv: 2201.10825. URL: https://arxiv.org/abs/2201.10825.

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.