Examining Games and a Way to Repair Them

Muhammad Najib

Department of Computer Science, University of Kaiserslautern

This is a summary of a series of works with Julian Gutierrez, Lewis Hammond, Anthony W. Lin, Giuseppe Perelli, and Mike Wooldridge.

The works published in IJCAI 2019, CONCUR 2019, AIJ, KR 2021.

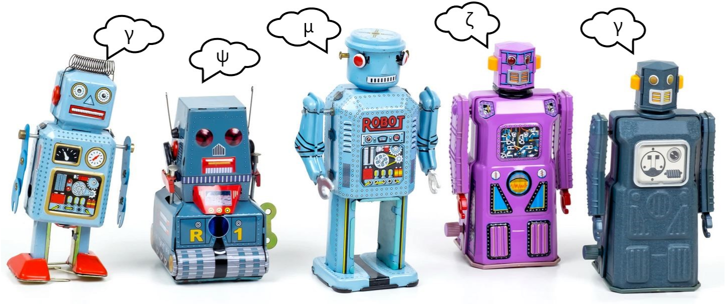
Structure

1. Examining
   - Correctness
   - Tractability
   - Cooperation and Probability
2. Repairing

Part I: Examining

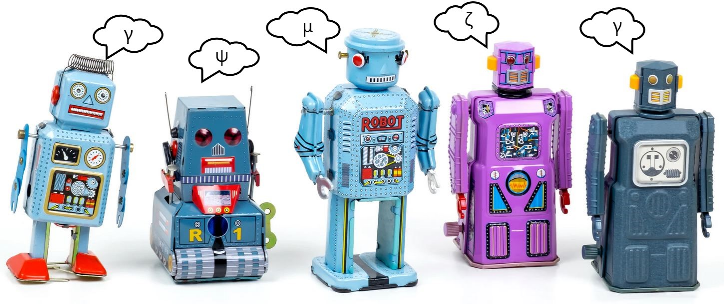
How should we define correctness in MAS?

How Should We Define Multi-Agent Systems Correctness?



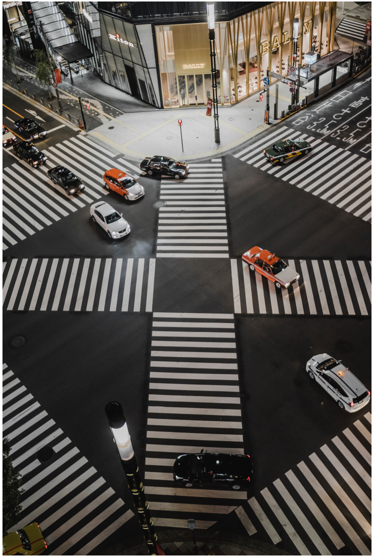
Classical notion of correctness ignores agents preferences

How Should We Define Multi-Agent Systems Correctness?



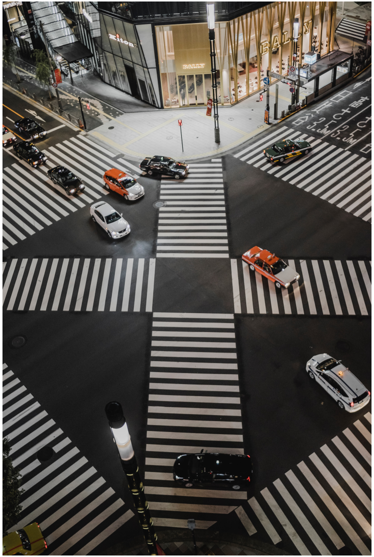
Correctness with respect to rational choices of agents

Not all behaviours are equal, but some are more unequal than others



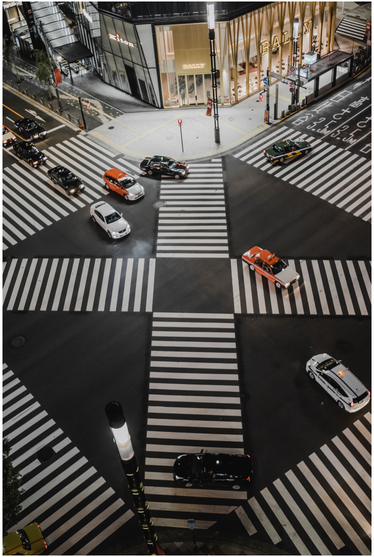
- Autonomous cars crossing an intersection

Not all behaviours are equal, but some are more unequal than others



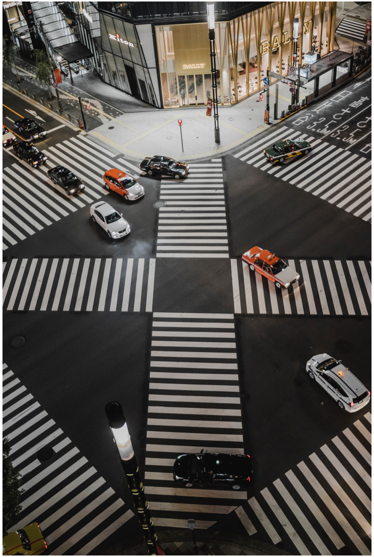
- Autonomous cars crossing an intersection
- Most of them (are expected to) cross without crashing with each other

Not all behaviours are equal, but some are more unequal than others



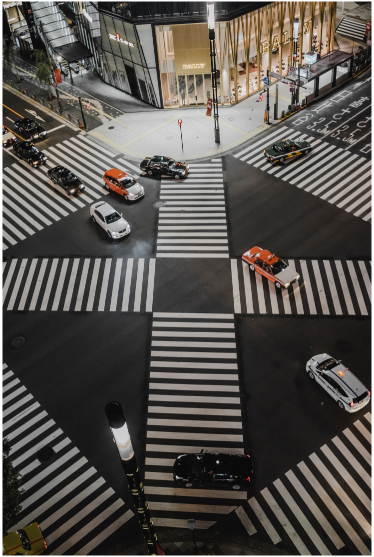
- Autonomous cars crossing an intersection
- Most of them (are expected to) cross without crashing with each other
- Cross and crash is also a *possible* behaviour of the system

Not all behaviours are equal, but some are more unequal than others



- Autonomous cars crossing an intersection
- Most of them (are expected to) cross without crashing with each other
- Cross and crash is also a *possible* behaviour of the system
- But cross and crash is not a *rational* behaviour

Not all behaviours are equal, but some are more unequal than others



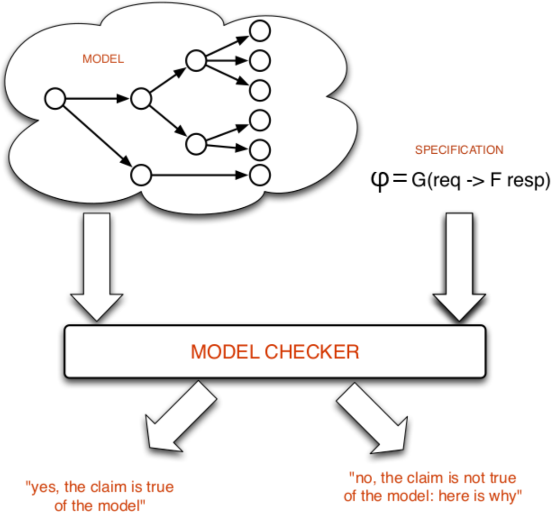
- Autonomous cars crossing an intersection
- Most of them (are expected to) cross without crashing with each other
- Cross and crash is also a *possible* behaviour of the system
- But cross and crash is not a *rational* behaviour
- They would rather do something else (not crash), thus it's not a stable behaviour

Multi-Agent Systems Correctness

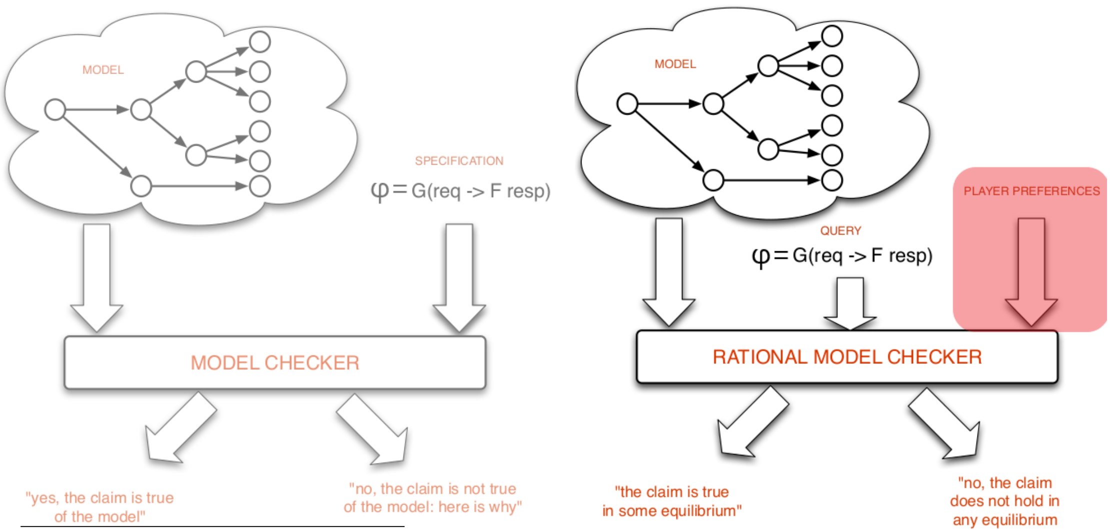
How do we define correctness in MAS?

- Is the system correct with respect to the set of stable behaviours?
- Stable behaviours in a group of *intelligent* agents $\Rightarrow$ game theory
- Turn MAS into multi-player game

From Verification to Rational Verification



From Verification to Rational Verification[1]



[1] M. Wooldridge et al. "Rational Verification: From Model Checking to Equilibrium Checking". In: *AAAI*. 2016, pp. 4184–4191.

- Game $\mathcal{G}$, each Player i is associated with a LTL goal γ_i
- Each player chooses a strategy; resolves non-deteminism.
- A LTL property φ

LTL Game

A multi-player LTL game is a tuple $\mathcal{G}_{\mathsf{LTL}} = (\mathcal{M}, \lambda, (\gamma_i)_{i \in \mathsf{N}})$

- $\mathcal{M} = (\mathsf{N}, (\mathsf{Ac}_i)_{i \in \mathsf{N}}, \mathsf{St}, s_0, \mathsf{tr})$ is a concurrent game arena (CGA) [a],
- γ_i is the LTL goal for player i.
- $\lambda : \mathsf{St} \to 2^{\mathsf{AP}}$ is a labelling function

[a] As usual: N agents; Ac_i actions of player i; St states; s_0 initial state; tr transition function.

Useful Games

A (2-player) *parity* game is a tuple $H = (V_0, V_1, E, \alpha)$

- zero-sum turn-based
- $V = V_0 \cup V_1$
- $E \subseteq V \times V$
- $\alpha : V \to \mathbb{N}$ is a labelling priority function

Player 0 wins if the smallest priority that occurs infinitely often in the infinite play is even. Otherwise, player 1 wins. Can be solved in NP $\cap$ coNP[a].

[a] Marcin Jurdziński. "Deciding the winner in parity games is in UP $\cap$ co-UP". In: *Information Processing Letters* (1998).

Useful Games

A multi-player *parity* game is a tuple $\mathcal{G}_{\mathsf{PAR}} = (\mathcal{M}, (\alpha_i)_{i \in \mathsf{N}})$

- $\mathcal{M} = (\mathsf{N}, (\mathsf{Ac}_i)_{i \in \mathsf{N}}, \mathsf{St}, s_0, \mathsf{tr})$ is a concurrent game arena (CGA) [2],
- $\alpha_i : \mathsf{St} \to \mathbb{N}$ is the goal of player i, given as a priority function over St.

[2] As usual: N agents; Ac_i actions of player i; St states; s_0 initial state; tr transition function.

Strategies and Plays

Strategy

Finite state machine $\sigma_i = \langle S_i, s_i^0, \delta_i, \tau_i \rangle$

- S_i, internal state (s_i^0 initial state);
- $\delta_i : S_i \times \mathrm{Ac} \to S_i$ internal transition function;
- $\tau_i : S_i \to \mathrm{Ac}_i$ action function.

A strategy is a recipe for the agent prescribing the action to take at every time-step of the game execution. A strategy profile $\vec{\sigma} = \langle \sigma_1, \ldots, \sigma_{\mathrm{N}} \rangle$ assigns a strategy to each agent in the arena.

Play

Given a strategy assigned to every agent in A, denoted $\vec{\sigma}$, there is a unique possible execution $\pi(\vec{\sigma})$ called play.

Note that plays can only be ultimately periodic.

Payoff Function

Let w_i be γ_i if $\mathcal{G}$ is an LTL game, and be α_i if $\mathcal{G}$ is a Parity game. For a strategy profiles $\vec{\sigma}$ in $\mathcal{G}$, we have

$$\mathsf{pay}_i(\pi(\vec{\sigma})) = \begin{cases} 1, & \text{if } \pi(\vec{\sigma}) \models w_i \\ 0, & \text{otherwise} \end{cases}$$

Nash Equilibria

Payoff Function

Let w_i be γ_i if $\mathcal{G}$ is an LTL game, and be α_i if $\mathcal{G}$ is a Parity game. For a strategy profiles $\vec{\sigma}$ in $\mathcal{G}$, we have

$$\mathsf{pay}_i(\pi(\vec{\sigma})) = \begin{cases} 1, & \text{if } \pi(\vec{\sigma}) \models w_i \\ 0, & \text{otherwise} \end{cases}$$

Nash Equilibrium

For a game $\mathcal{G}$, a strategy profile $\vec{\sigma}$ is a *Nash equilibrium* of $\mathcal{G}$ if, for every player i and strategy $\sigma'_i \in \mathsf{Str}_i$, we have

$$\mathsf{pay}_i(\pi(\vec{\sigma})) \geq \mathsf{pay}_i(\pi((\vec{\sigma}_{-i}, \sigma'_i))) \ .$$

i.e., **no player can benefit by changing its strategy unilaterally.**

E-Nash

Given: Game $\mathcal{G}$, temporal property φ.

Quest: Is there any Nash Equilibrium $\vec{\sigma}$ in $\mathcal{G}$ such that $\pi(\vec{\sigma}) \models \varphi$?

Rational Verification: Decision Problems

E-Nash

Given: Game $\mathcal{G}$, temporal property φ.

Quest: Is there any Nash Equilibrium $\vec{\sigma}$ in $\mathcal{G}$ such that $\pi(\vec{\sigma}) \models \varphi$?

A-Nash

Given: Game $\mathcal{G}$, temporal property φ.

Quest: Does $\pi(\vec{\sigma}) \models \varphi$ hold for every Nash Equilibrium $\vec{\sigma}$ in $\mathcal{G}$?

Rational Verification: Decision Problems

E-Nash

Given: Game $\mathcal{G}$, temporal property φ.

Quest: Is there any Nash Equilibrium $\vec{\sigma}$ in $\mathcal{G}$ such that $\pi(\vec{\sigma}) \models \varphi$?

A-Nash

Given: Game $\mathcal{G}$, temporal property φ.

Quest: Does $\pi(\vec{\sigma}) \models \varphi$ hold for every Nash Equilibrium $\vec{\sigma}$ in $\mathcal{G}$?

Both decision problems above can be reduced to the following

Non-Emptiness

Given: Game $\mathcal{G}$.

Quest: Is there any Nash Equilibrium in $\mathcal{G}$?

NE Characterisation

Theorem (NE characterisation)

Let $NE(\mathcal{G})$ be the set of Nash equilibria in $\mathcal{G}$. A strategy profile $\vec{\sigma} \in NE(\mathcal{G})$

if and only if

the path $\pi = \pi(\vec{\sigma})$ is such that, for every $k \in \mathbb{N}$, the pair $(s_k, \vec{a}^k)$ of the k-th position of π is punishing secure [3] *for every $j \in Lose(\pi)$.* [4] *Where $\vec{a}^k = \langle a_1, ..., a_n \rangle$ is an action profile at k.*

Along π, no player j can unilaterally get its goal γ_j achieved.

[3] Punishing secure: agent j does not have a strategy σ'_j that wins against $\vec{\sigma}_{-j}$, i.e. $\pi(\vec{\sigma}_{-j}, \sigma'_j) \models \gamma_j$.

[4] Here $\text{Lose}(\pi) = \{j \in \text{N} : \pi \not\models \gamma_j\}$ are the agents that are not satisfied over π.

NE Characterisation via Local Reasoning

- Memory is needed to satisfy LTL goal
- Memory is NOT necessary for (2-player) parity games (memoryless/positional determinacy)
- Reason locally by converting each γ_i into deterministic parity word automaton (DPW) $\mathcal{A}_i = \langle 2^{\mathsf{AP}}, Q, q^0, \rho, \alpha \rangle$.
- Then build $\mathcal{G}_{\mathsf{LTL}} = (\mathcal{M}, \lambda, (\gamma_i)_{i \in \mathsf{N}})$ into $\mathcal{G}_{\mathsf{PAR}} = (\mathcal{M}', (\alpha'_i)_{i \in \mathsf{N}})$, where $\mathcal{M}' = (\mathsf{N}, (\mathsf{Ac}_i)_{i \in \mathsf{N}}, \mathsf{St}', s'_0, \mathsf{tr}')$ and $(\alpha'_i)_{i \in \mathsf{N}}$:
  - $\mathsf{St}' = \mathsf{St} \times \bigtimes_{i \in \mathsf{N}} Q_i$ and $s'_0 = (s_0, q^0_1, \ldots, q^0_n)$;
  - for each state $(s, q_1, \ldots, q_n) \in \mathsf{St}'$ and action profile $\vec{a}$, $\mathsf{tr}'((s, q_1, \ldots, q_n), \vec{a}) = (\mathsf{tr}(s, \vec{a}), \rho_1(q_1, \lambda(s)), \ldots, \rho_n(q_n, \lambda(s)))$;
  - $\alpha'_i(s, q_1, \ldots q_n) = \alpha_i(q_i)$.

Invariances[5]

Lemma (Goal Invariance)

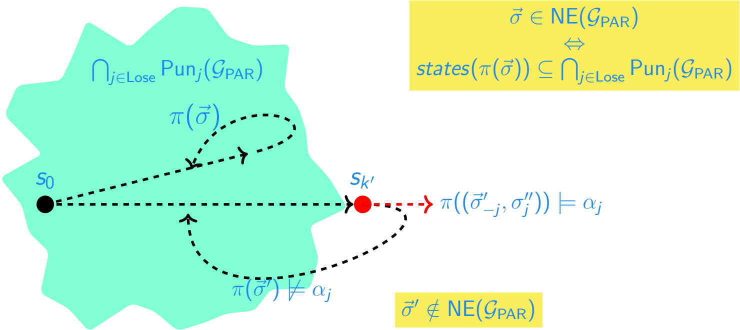
Let $\mathcal{G}_{\mathsf{LTL}}$ be an LTL game and $\mathcal{G}_{PAR}$ its associated Parity game. Then, for every strategy profile $\vec{\sigma}$ and player i, it is the case that $\pi(\vec{\sigma}) \models \gamma_i$ in $\mathcal{G}_{\mathsf{LTL}}$ if and only if $\pi(\vec{\sigma}) \models \alpha_i$ in $\mathcal{G}_{PAR}$.

Theorem (NE Invariance)

Let $\mathcal{G}_{\mathsf{LTL}}$ be an LTL game and $\mathcal{G}_{PAR}$ its associated Parity game. Then, $NE(\mathcal{G}_{\mathsf{LTL}}) = NE(\mathcal{G}_{PAR})$.

[5] Julian Gutierrez et al. "Automated temporal equilibrium analysis: Verification and synthesis of multi-player games". In: *Artificial Intelligence* (2020).

Visualising NE Characterisation



$\bigcap_{j \in \mathsf{Lose}} \mathsf{Pun}_j(\mathcal{G}_{\mathsf{PAR}})$ is the punishing region for Lose

Computing Punishing Region

For a $\mathcal{G}_{\mathsf{PAR}}$ and a (to-be-punished) player j. We turn $\mathcal{G}_{\mathsf{PAR}}$ into a 2-player zero-sum parity game $H_j = (V_0, V_1, E, \alpha)$ between player j (Player 1) and (coalition) player N_{-j} (Player 0). Circular states are in V_0.

$$s_1 \xrightarrow{(\vec{a}_{-j},\, a_j)} s_2 \qquad\qquad s_1 \longrightarrow \boxed{(s_1, \vec{a}_{-j})} \longrightarrow s_2$$

punishing region for $\mathsf{Lose} = \bigcap_{j \in \mathsf{Lose}} \mathsf{Pun}_j(\mathcal{G}_{\mathsf{PAR}})$

Computing Punishing Region

For a $\mathcal{G}_{\mathsf{PAR}}$ and a (to-be-punished) player j. We turn $\mathcal{G}_{\mathsf{PAR}}$ into a 2-player zero-sum parity game $H_j = (V_0, V_1, E, \alpha)$ between player j (Player 1) and (coalition) player N_{-j} (Player 0). Circular states are in V_0.

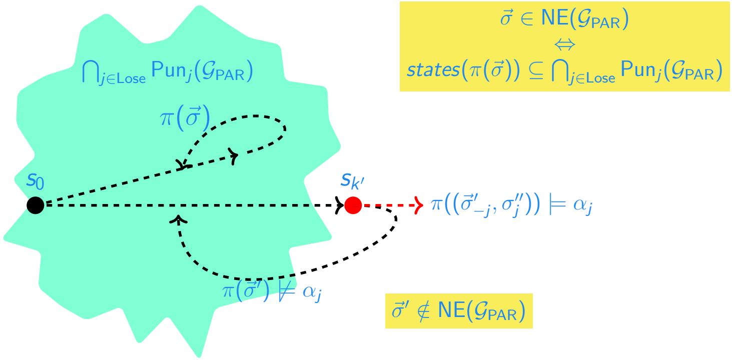
$$s_1 \xrightarrow{(\vec{a}_{-j}, a_j)} s_2 \qquad\qquad s_1 \longrightarrow \boxed{(s_1, \vec{a}_{-j})} \longrightarrow s_2$$

punishing region for $\mathsf{Lose} = \bigcap_{j \in \mathsf{Lose}} \mathsf{Pun}_j(\mathcal{G}_{\mathsf{PAR}})$

Corollary

Computing $Pun_i(\mathcal{G}_{PAR})$ can be done in polynomial time with respect to the size of the underlying graph of the game $\mathcal{G}_{PAR}$ and exponential in the size of the priority function α_i, that is, to the size of the range of α_i. Moreover, there is a memoryless strategy $\vec{\sigma}_i$ that is a punishment against player i in every state $s \in Pun_i(\mathcal{G}_{PAR})$.

Finding NE Run



How do we compute $\pi(\vec{\sigma})$? Is there such run $\pi(\vec{\sigma})$ inside the punishing region?

Finding NE Run

- $\pi(\vec{\sigma})$ must be accepting for each $\alpha_i, i \in \mathsf{Win} = \mathsf{N} \setminus \mathsf{Lose}$.
- Solve emptiness problem of DPWs intersection $\bigtimes_{i \in \mathsf{Win}} \mathcal{A}^i$
- Intersection of DPWs might involve exponential blowup
- Each parity condition $\alpha = (F_1, \ldots, F_n)$ is a Streett condition $((E_1, C_1), \ldots, (E_m, C_m))$ with $m = \lceil \frac{n}{2} \rceil$ and $(E_i, C_i) = (F_{2i+1}, \bigcup_{j \leq i} F_{2j})$, for each $0 \leq i \leq m$
- Intersection of (Deterministic Streett Word Automata) DSWs $\bigtimes_{i \in \mathsf{Win}} \mathcal{S}_i$ and nonemptiness check can be done in polynomial time

The Procedure

1. $\mathcal{G}_{\text{LTL}} \Rightarrow \mathcal{G}_{\text{PAR}}$
2. For each $\text{Win} \subseteq \text{N}$ do:
   2.1 Compute punishing region $\bigcap_{j \in \text{Lose}} \text{Pun}_j(\mathcal{G}_{\text{PAR}})$
   2.2 Construct DSW $\bigtimes_{i \in \text{Win}} \mathcal{S}_i$
   2.3 If $\mathcal{L}(\bigtimes_{i \in \text{Win}} \mathcal{S}_i) \neq \varnothing$ then return "YES"
3. Return "NO"

- Step 1 can be done in 2EXPTIME: the number of states is doubly exponential in the size of LTL goals, but priority functions $(\alpha_i)_{i \in \text{N}}$ is only singly exponential.
- Step 2 at most executed exponential in the number of players
- Step 2.1 is polynomial in the number of states and exponential in the number of priorities
- Step 2.2 and 2.3 are both polynomial in the number of states
- Overall we have 2EXPTIME procedure.

The Procedure

1. $\mathcal{G}_{\mathsf{LTL}} \Rightarrow \mathcal{G}_{\mathsf{PAR}}$
2. For each $\mathsf{Win} \subseteq \mathsf{N}$ do:
   2.1 Compute punishing region $\bigcap_{j \in \mathsf{Lose}} \mathsf{Pun}_j(\mathcal{G}_{\mathsf{PAR}})$
   2.2 Construct DSW $\bigtimes_{i \in \mathsf{Win}} \mathcal{S}_i$
   2.3 If $\mathcal{L}(\bigtimes_{i \in \mathsf{Win}} \mathcal{S}_i) \neq \varnothing$ then return "YES"
3. Return "NO"

- Step 1 can be done in 2EXPTIME: the number of states is doubly exponential in the size of LTL goals, but priority functions $(\alpha_i)_{i \in \mathsf{N}}$ is only singly exponential.
- Step 2 at most executed exponential in the number of players
- Step 2.1 is polynomial in the number of states and exponential in the number of priorities
- Step 2.2 and 2.3 are both polynomial in the number of states
- Overall we have 2EXPTIME procedure.

The Procedure

1. $\mathcal{G}_{\mathsf{LTL}} \Rightarrow \mathcal{G}_{\mathsf{PAR}}$
2. For each $\mathsf{Win} \subseteq \mathsf{N}$ do:
   2.1 Compute punishing region $\bigcap_{j \in \mathsf{Lose}} \mathsf{Pun}_j(\mathcal{G}_{\mathsf{PAR}})$
   2.2 Construct DSW $\bigtimes_{i \in \mathsf{Win}} \mathcal{S}_i$
   2.3 If $\mathcal{L}(\bigtimes_{i \in \mathsf{Win}} \mathcal{S}_i) \neq \varnothing$ then return "YES"
3. Return "NO"

- Step 1 can be done in 2EXPTIME: the number of states is doubly exponential in the size of LTL goals, but priority functions $(\alpha_i)_{i \in \mathsf{N}}$ is only singly exponential.
- Step 2 at most executed exponential in the number of players
- Step 2.1 is polynomial in the number of states and exponential in the number of priorities
- Step 2.2 and 2.3 are both polynomial in the number of states
- Overall we have 2EXPTIME procedure.

The Procedure

1. $\mathcal{G}_{\mathsf{LTL}} \Rightarrow \mathcal{G}_{\mathsf{PAR}}$
2. For each $\mathsf{Win} \subseteq \mathsf{N}$ do:
   - 2.1 Compute punishing region $\bigcap_{j \in \mathsf{Lose}} \mathsf{Pun}_j(\mathcal{G}_{\mathsf{PAR}})$
   - 2.2 Construct DSW $\bigtimes_{i \in \mathsf{Win}} \mathcal{S}_i$
   - 2.3 If $\mathcal{L}(\bigtimes_{i \in \mathsf{Win}} \mathcal{S}_i) \neq \varnothing$ then return "YES"
3. Return "NO"

- Step 1 can be done in 2EXPTIME: the number of states is doubly exponential in the size of LTL goals, but priority functions $(\alpha_i)_{i \in \mathsf{N}}$ is only singly exponential.
- Step 2 at most executed exponential in the number of players
- Step 2.1 is polynomial in the number of states and exponential in the number of priorities
- Step 2.2 and 2.3 are both polynomial in the number of states
- Overall we have 2EXPTIME procedure.

The Procedure

1. $\mathcal{G}_{\mathsf{LTL}} \Rightarrow \mathcal{G}_{\mathsf{PAR}}$
2. For each $\mathrm{Win} \subseteq \mathrm{N}$ do:
   2.1 Compute punishing region $\bigcap_{j \in \mathrm{Lose}} \mathrm{Pun}_j(\mathcal{G}_{\mathsf{PAR}})$
   2.2 Construct DSW $\bigtimes_{i \in \mathrm{Win}} \mathcal{S}_i$
   2.3 If $\mathcal{L}(\bigtimes_{i \in \mathrm{Win}} \mathcal{S}_i) \neq \varnothing$ then return “YES”
3. Return “NO”

- Step 1 can be done in 2EXPTIME: the number of states is doubly exponential in the size of LTL goals, but priority functions $(\alpha_i)_{i \in \mathrm{N}}$ is only singly exponential.
- Step 2 at most executed exponential in the number of players
- Step 2.1 is polynomial in the number of states and exponential in the number of priorities
- Step 2.2 and 2.3 are both polynomial in the number of states
- Overall we have 2EXPTIME procedure.

The Procedure

1. $\mathcal{G}_{\mathsf{LTL}} \Rightarrow \mathcal{G}_{\mathsf{PAR}}$
2. For each $\mathsf{Win} \subseteq \mathsf{N}$ do:
   2.1 Compute punishing region $\bigcap_{j \in \mathsf{Lose}} \mathsf{Pun}_j(\mathcal{G}_{\mathsf{PAR}})$
   2.2 Construct DSW $\bigtimes_{i \in \mathsf{Win}} \mathcal{S}_i$
   2.3 If $\mathcal{L}(\bigtimes_{i \in \mathsf{Win}} \mathcal{S}_i) \neq \varnothing$ then return "YES"
3. Return "NO"

- Step 1 can be done in 2EXPTIME: the number of states is doubly exponential in the size of LTL goals, but priority functions $(\alpha_i)_{i \in \mathsf{N}}$ is only singly exponential.
- Step 2 at most executed exponential in the number of players
- Step 2.1 is polynomial in the number of states and exponential in the number of priorities
- Step 2.2 and 2.3 are both polynomial in the number of states
- Overall we have 2EXPTIME procedure.

Rational Verification: Complexity

Theorem (Complexity)

For the case of both the specification φ and the agents goals γ_i expressed as LTL formulas, E-Nash and A-Nash are 2EXPTIME-Complete.[6]

[6] Wooldridge et al., "Rational Verification: From Model Checking to Equilibrium Checking"; Julian Gutierrez, Paul Harrenstein, and Michael J. Wooldridge. "From model checking to equilibrium checking: Reactive modules for rational verification". In: *Artificial Intelligence* 248 (2017), pp. 123–157.

EVE (Equilibrium Verification Environment)

- Simple Reactive Modules Language (SRML)[7] as modelling language
- Supports general-sum multi-player LTL games, bisimulation-invariant strategies, and perfect recall.
- Supports Non-emptiness, E-Nash, and A-Nash
- Synthesise strategies
- **Open-source:** `https://github.com/eve-mas/eve-parity`
- **EVE Online:** http://eve.cs.ox.ac.uk/

[7] Based on the Reactive Modules language used by PRISM and MOCHA.

Chapter 2: Tractability[8]

- 2EXPTIME is rather slow
- What can we do to improve?
- Use different goals and properties: GR(1) and mean-payoff value

[8] Julian Gutierrez et al. "On Computational Tractability for Rational Verification". In: *IJCAI*. 2019, pp. 329–335.

GR(1)[9]

The language of *General Reactivity of rank 1*, denoted GR(1), is the fragment of LTL of formulae written in the following form:

$$(\mathbf{GF}\psi_1 \wedge \ldots \wedge \mathbf{GF}\psi_m) \rightarrow (\mathbf{GF}\varphi_1 \wedge \ldots \wedge \mathbf{GF}\varphi_n),$$

each ψ_i and φ_i is a Boolean combination of atomic propositions.

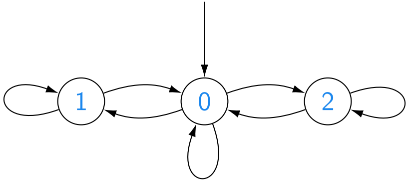
$$(\mathbf{GF}\mathit{req}_1 \wedge \mathbf{GF}\mathit{req}_2) \rightarrow \mathbf{GF}\mathit{ack}$$

[9] Roderick Bloem et al. "Synthesis of Reactive(1) designs". In: *J. Comput. Syst. Sci.* 78.3 (2012), pp. 911–938.

Mean-payoff value

For an infinite sequence $\beta \in \mathbb{R}^\omega$ of real numbers, let $\mathrm{mp}(\beta)$ be the *mean-payoff* value of β, defined as follows:

$$\mathrm{mp}(\beta) = \lim_{n\to\infty} \inf \frac{1}{n} \sum_{i=0}^{n-1} \beta[i]$$



$$\beta^1 = 00000000000\ldots \qquad \mathrm{mp}(\beta^1) = 0$$

$$\beta^2 = 01010101010\ldots \qquad \mathrm{mp}(\beta^2) = 0.5$$

$$\beta^3 = 01020102010\ldots \qquad \mathrm{mp}(\beta^3) = 3/4$$

Games

A multi-player $\mathsf{GR}(1)$ game is a tuple $\mathcal{G}_{\mathsf{GR}(1)} = \langle \mathcal{M}, (\gamma_i)_{i \in \mathsf{N}}, \lambda \rangle$

- $\mathcal{M} = \langle \mathsf{N}, \mathsf{Ac}, \mathsf{St}, s_0, \mathsf{tr} \rangle$ is an arena,
- γ_i is the $\mathsf{GR}(1)$ goal for player i.

A multi-player mp game is a tuple $\mathcal{G}_{\mathsf{mp}} = \langle \mathcal{M}, (\mathsf{w}_i)_{i \in \mathsf{N}}, \lambda \rangle$,

- $\mathcal{M} = \langle \mathsf{N}, \mathsf{Ac}, \mathsf{St}, s_0, \mathsf{tr} \rangle$ is an arena
- $\mathsf{w}_i : \mathsf{St} \to \mathbb{Z}$ maps states to integer numbers, for each player i

Cases

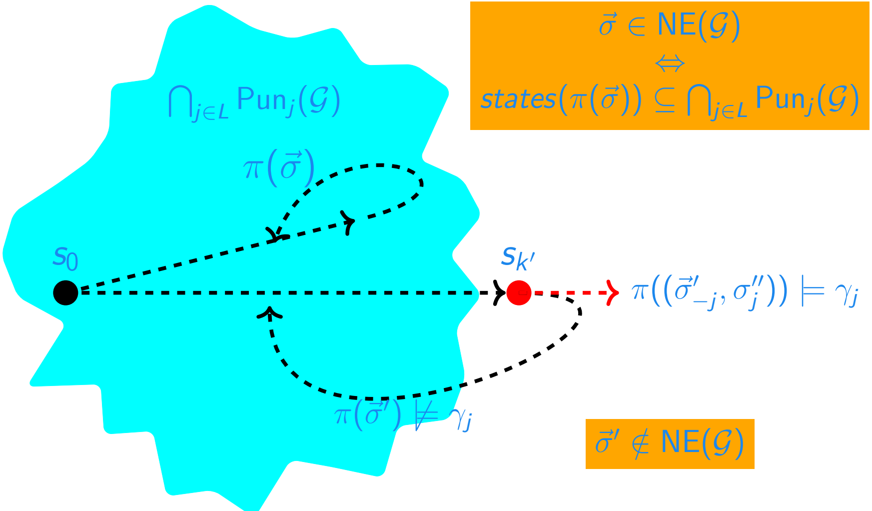
E-Nash

Given: Game $\mathcal{G}$, temporal property φ.
Quest: Is there any Nash Equilibrium $\vec{\sigma}$ in $\mathcal{G}$ such that $\pi(\vec{\sigma}) \models \varphi$?

	γ_i	φ	E-Nash
	LTL	LTL	2EXPTIME-complete
GR(1) games	GR(1)	LTL	?
GR(1) games	GR(1)	GR(1)	?
mp games	mp	LTL	?
mp games	mp	GR(1)	?

E-Nash in GR(1) games: NE characterisation

Theorem (NE characterisation)



E-Nash in GR(1) games: Computing punishment regions

Theorem (Computing $\mathbf{Pun}_j(\mathcal{G})$)

For a given GR(1) game $\mathcal{G}$, computing $\mathsf{Pun}_j(\mathcal{G})$ of player j can be done in polynomial time with respect to the size of both $\mathcal{G}$ and γ_j.

E-Nash in GR(1) games: the procedure

1. Guess a set $\mathsf{Win} \subseteq \mathsf{N}$ of winners;
2. For each player $j \in \mathsf{Lose} = \mathsf{N} \setminus \mathsf{Win}$, a loser in the game, compute its punishment region $\mathsf{Pun}_j(\mathcal{G})$;
3. Find desired path $\pi(\vec{\sigma})$ consisting of states in $\bigcap_{j \in \mathsf{Lose}} \mathsf{Pun}_j(\mathcal{G})$. Any deviation by player j must remain inside $\mathsf{Pun}_j(\mathcal{G})$, that is, a path $\pi(\vec{\sigma})$ satisfying the following three conditions:
    - $\mathit{states}(\pi(\vec{\sigma})) \subseteq \bigcap_{j \in \mathsf{Lose}} \mathsf{Pun}_j(\mathcal{G})$
    - $\mathit{states}(\pi(\vec{\sigma}_{-j}, \sigma'_j)) \subseteq \mathsf{Pun}_j(\mathcal{G})$, for every $j \in \mathsf{Lose}$ and σ'_j of j
    - $\pi(\vec{\sigma}) \models \varphi \wedge \bigwedge_{i \in \mathsf{Win}} \gamma_i$

E-Nash in GR(1) games: the procedure

1. Guess a set $\mathsf{Win} \subseteq \mathsf{N}$ of winners;
2. For each player $j \in \mathsf{Lose} = \mathsf{N} \setminus \mathsf{Win}$, a loser in the game, compute its punishment region $\mathsf{Pun}_j(\mathcal{G})$;
3. Find desired path $\pi(\vec{\sigma})$ consisting of states in $\bigcap_{j \in \mathsf{Lose}} \mathsf{Pun}_j(\mathcal{G})$. Any deviation by player j must remain inside $\mathsf{Pun}_j(\mathcal{G})$, that is, a path $\pi(\vec{\sigma})$ satisfying the following three conditions:
   - $\mathit{states}(\pi(\vec{\sigma})) \subseteq \bigcap_{j \in \mathsf{Lose}} \mathsf{Pun}_j(\mathcal{G})$
   - $\mathit{states}(\pi(\vec{\sigma}_{-j}, \sigma'_j)) \subseteq \mathsf{Pun}_j(\mathcal{G})$, for every $j \in \mathsf{Lose}$ and σ'_j of j
   - $\pi(\vec{\sigma}) \models \varphi \wedge \bigwedge_{i \in \mathsf{Win}} \gamma_i$

Complexities for GR(1) and LTL specifications:

- If φ is a GR(1) specification: FPT
- If φ is an LTL specification: PSPACE

E-Nash in mp games: NE characterisation

Theorem (NE characterisation)

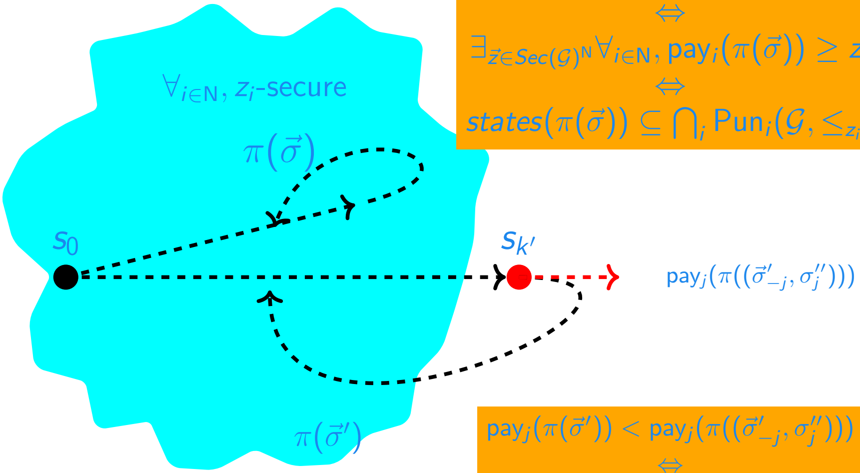
For every mp *game* $\mathcal{G}$ *and ultimately periodic path* $\pi = (s_0, \vec{a}^0), (s_1, \vec{a}^1), \ldots$, *the following are equivalent*

1. *There is* $\vec{\sigma} \in NE(\mathcal{G})$ *such that* $\pi = \pi(\vec{\sigma})$;
2. *There exists* $\vec{z} \in \mathbb{R}^N$, *where* $z_i \in \{pun_i(s) : s \in St\}$ *such that, for every* $i \in N$
   - 2.1 $z_i \leq pay_i(\pi)$, *and*
   - 2.2 *for all* $k \in \mathbb{N}$, *the pair* $(s_k, \vec{a}^k)$ *is* z_i*-secure for* i.

Along π, no player i can unilaterally get a payoff greater than z_i.

E-Nash in mp games: NE characterisation

$$\vec{\sigma} \in \mathsf{NE}(\mathcal{G})$$
$$\Leftrightarrow$$
$$\exists_{\vec{z} \in Sec(\mathcal{G})^{\mathsf{N}}} \forall_{i \in \mathsf{N}}, \mathsf{pay}_i(\pi(\vec{\sigma})) \geq z_i$$
$$\Leftrightarrow$$
$$states(\pi(\vec{\sigma})) \subseteq \bigcap_i \mathsf{Pun}_i(\mathcal{G}, \leq_{z_i})$$



$$\mathsf{pay}_j(\pi(\vec{\sigma}')) < \mathsf{pay}_j(\pi((\vec{\sigma}'_{-j}, \sigma''_j)))$$
$$\Leftrightarrow$$
$$\vec{\sigma}' \notin \mathsf{NE}(\mathcal{G})$$

E-Nash in mp games: the procedure

1. Guess a vector $\vec{z} \in \mathbb{R}^{\mathrm{N}}$ of values, each being a punishment value for a player i
2. For each i, compute its z_i-punishment region $\mathrm{Pun}_i(\mathcal{G}, \leq_{z_i})$;
3. Find (u.p.) path $\pi(\vec{\sigma})$ consisting of states in $\bigcap_i \mathrm{Pun}_i(\mathcal{G}, \leq_{z_i})$. Any deviation by player i must remain inside $\mathrm{Pun}_i(\mathcal{G}, \leq_{z_i})$, that is, an ultimately periodic path $\pi(\vec{\sigma})$ satisfying that:
   - $\mathit{states}(\pi(\vec{\sigma})) \subseteq \bigcap_i \mathrm{Pun}_i(\mathcal{G}, \leq_{z_i})$
   - $\mathit{states}(\pi(\vec{\sigma}_{-i}, \sigma'_i)) \subseteq \mathrm{Pun}_i(\mathcal{G}, \leq_{z_i})$, for every i and σ'_i of i
   - $\pi(\vec{\sigma}) \models \varphi$ and $\forall_i, \mathrm{pay}_i(\pi(\vec{\sigma})) \geq z_i$

E-Nash in mp games: the procedure

1. Guess a vector $\vec{z} \in \mathbb{R}^{\mathrm{N}}$ of values, each being a punishment value for a player i
2. For each i, compute its z_i-punishment region $\mathrm{Pun}_i(\mathcal{G}, \leq_{z_i})$;
3. Find (u.p.) path $\pi(\vec{\sigma})$ consisting of states in $\bigcap_i \mathrm{Pun}_i(\mathcal{G}, \leq_{z_i})$. Any deviation by player i must remain inside $\mathrm{Pun}_i(\mathcal{G}, \leq_{z_i})$, that is, an ultimately periodic path $\pi(\vec{\sigma})$ satisfying that:
   - $\mathit{states}(\pi(\vec{\sigma})) \subseteq \bigcap_i \mathrm{Pun}_i(\mathcal{G}, \leq_{z_i})$
   - $\mathit{states}(\pi(\vec{\sigma}_{-i}, \sigma'_i)) \subseteq \mathrm{Pun}_i(\mathcal{G}, \leq_{z_i})$, for every i and σ'_i of i
   - $\pi(\vec{\sigma}) \models \varphi$ and $\forall_i, \mathrm{pay}_i(\pi(\vec{\sigma})) \geq z_i$

Complexities for GR(1) and LTL specifications:

- If φ is a GR(1) specification: NP-complete
- If φ is an LTL specification: PSPACE-complete

Complexity Results

γ_i	φ	E-NASH
LTL	LTL	2EXPTIME-complete
GR(1)	LTL	PSPACE-complete
GR(1)	GR(1)	FPT
mp	LTL	PSPACE-complete
mp	GR(1)	NP-complete

- NON-EMPTINESS (E-NASH when $\varphi = \top$):
  - LTL games: 2EXPTIME-complete
  - GR(1) games: FPT
  - mp games: NP-complete
- A-NASH: 2EXPTIME, PSPACE, FPT, PSPACE, coNP.

Chapter 3: Cooperation and Probability[10]

- Players can make a binding agreements and form coalitions
- Coalitions can collectively achieve goals
- Cooperative games
- Solution concept: **Core**

[10] Julian Gutierrez et al. "Rational Verification for Probabilistic Systems". In: *KR*. to appear. 2021.

Rational Verification in Cooperative Games

- Game $\mathcal{G}$, each Player i is associated with a LTL goal γ_i
- Each player chooses a strategy; resolves non-deteminism.
- A LTL property φ

Rational Verification in Cooperative Games

- Game $\mathcal{G}$, each Player i is associated with a LTL goal γ_i
- Each player chooses a strategy; resolves non-deteminism.
- A LTL property φ

E-Core

Is there any *core* $\vec{\sigma}$ in $\mathcal{G}$ such that $\pi(\vec{\sigma}) \models \varphi$?

Rational Verification in Cooperative Games

- Game $\mathcal{G}$, each Player i is associated with a LTL goal γ_i
- Each player chooses a strategy; resolves non-deteminism.
- A LTL property φ

E-Core

Is there any *core* $\vec{\sigma}$ in $\mathcal{G}$ such that $\pi(\vec{\sigma}) \models \varphi$?

A-Core

Does $\pi(\vec{\sigma}) \models \varphi$ hold for every *core* $\vec{\sigma}$ in $\mathcal{G}$?

Rational Verification

- Game $\mathcal{G}$, each Player i is associated with a LTL goal γ_i
- A strategy profile $\vec{\sigma}$

Core-Membership

Is $\vec{\sigma}$ a core in the game $\mathcal{G}$?

Theorem (Complexity)

For the case of both the specification φ and the agents goals γ_i expressed as LTL *formulas, E-Core, A-Core, and Core-Membership are* 2EXPTIME-Complete.[11]

[11] Julian Gutierrez, Sarit Kraus, and Michael Wooldridge. "Cooperative Concurrent Games". In: *AAMAS*. 2019.

Complexity Map

	Non-Cooperative	**Cooperative**
E-(Nash/Core)	2EXPTIME-Complete	2EXPTIME-Complete
A-(Nash/Core)	2EXPTIME-Complete	2EXPTIME-Complete
(NE/Core)-Membership	PSPACE-Complete	2EXPTIME-Complete

Complexity Map

	Non-Cooperative	**Cooperative**
E-(Nash/Core)	2EXPTIME-Complete	2EXPTIME-Complete
A-(Nash/Core)	2EXPTIME-Complete	2EXPTIME-Complete
(NE/Core)-Membership	PSPACE-Complete	2EXPTIME-Complete

Without probabilistic behaviours...

A Case for Probabilistic Systems

- Real life scenarios often involve probabilities
- Probabilities grant us power, e.g., the dining philosopher problem

This Work

- Rational verification for probabilistic systems
- Cooperative and non-cooperative games
- Goals and specifications are LTL formulae
- Concurrent actions, infinite horizon, infinite memory
- *Qualitative setting:* almost-surely satisfaction

Concurrent Stochastic Games (CSGs)

Definition (CSG Arena)

A *concurrent stochastic game arena* (CSGA) is a tuple $\mathcal{M} = (\mathsf{N}, \mathsf{St}, s^0, (\mathsf{Ac}_i)_{i \in \mathsf{N}}, \mathsf{tr})$, where

- $\mathsf{tr} : \mathsf{St} \times \vec{\mathsf{Ac}} \to \mathsf{D}(\mathsf{St})$ is probabilistic transition function

Definition (CSG)

A *concurrent stochastic game* (CSG) is a tuple $\mathcal{G} = (\mathcal{M}, (\gamma_i)_{i \in \mathsf{N}}, \lambda)$, where $\mathcal{M}$ is a CSGA, γ_i is a LTL formula that represents the *goal* of player i, and $\lambda : \mathsf{St} \to 2^{\mathsf{AP}}$ a labelling function.

Concurrent Stochastic Parity Games (CSPGs)

Definition (CSG Arena)

A *concurrent stochastic game arena* (CSGA) is a tuple $\mathcal{M} = (\mathsf{N}, \mathsf{St}, s^0, (\mathsf{Ac}_i)_{i \in \mathsf{N}}, \mathsf{tr})$, where

- $\mathsf{tr} : \mathsf{St} \times \vec{\mathsf{Ac}} \to \mathsf{D}(\mathsf{St})$ is probabilistic transition function

Definition (CSPG)

A *concurrent stochastic parity game* (CSPG) is a tuple $\mathcal{G}_{\mathsf{PAR}} = (\mathcal{M}, (\alpha_i)_{i \in \mathsf{N}})$, where $\alpha_i : \mathsf{St} \to \mathbb{N}$ is the goal of player i, given as a priority function over the set of states St. A path π satisfies a priority function α, denoted by $\pi \models \alpha$, if the minimum number occuring infinitely often in the infinite sequence $\alpha(\pi_0)\alpha(\pi_1)\alpha(\pi_2)\ldots$ is even.

Strategies

Definition (Strategy)

A strategy for player i can be understood (abstractly) as a function $\sigma_i : \mathsf{St}^+ \to \mathsf{D}(\mathsf{Ac}_i)$ that assigns to every non-empty finite sequence of states a probability distribution over player i's set of actions.

Strategies

Definition (Strategy as Transducer)

a *strategy* in $\mathcal{G}$ for player i is a a transducer $\sigma_i = (Q_i, q_i^0, \delta_i, \tau_i)$

A strategy is

- *memoryless* if there exists a transducer encoding the strategy with $|Q_i| = 1$
- finite-memory if $|Q_i| < \infty$
- *deterministic* if $\tau_i : Q_i \times \mathrm{St} \to \mathrm{Ac}_i$, such that for every $q_i \in Q_i$ and every $s \in \mathrm{St}$, we have that $\tau_i(q_i, s) \in \mathrm{Ac}_i(s)$

Satisfaction Conditions

LTL goals:

- For a given game $\mathcal{G}$ and a strategy profile $\vec{\sigma}$, a formula φ is said to be *almost-surely* satisfied, denoted $\vec{\sigma} \models \mathsf{AS}(\varphi)$, iff, $\mathsf{Pr}_{\mathcal{C}_{\vec{\sigma}}}(\{\pi \in \mathrm{Paths}(\mathcal{C}_{\vec{\sigma}}, s^0) : \pi \models \varphi\}) = 1$.
- φ is satisfied with *non-zero* probability, denoted $\vec{\sigma} \models \mathsf{NZ}(\varphi)$ iff $\mathsf{Pr}_{\mathcal{C}_{\vec{\sigma}}}(\{\pi \in \mathrm{Paths}(\mathcal{C}_{\vec{\sigma}}, s^0) : \pi \models \varphi\}) > 0$.
- $\mathsf{NZ}(\varphi) \equiv \neg\mathsf{AS}(\neg\varphi)$

Parity goals:

- $\vec{\sigma} \models \mathsf{AS}(\alpha)$ if and only if $\mathsf{Pr}_{\mathcal{C}_{\vec{\sigma}}}(\{\pi \in \mathrm{Paths}(\mathcal{C}_{\vec{\sigma}}, s^0) : \pi \models \alpha\}) = 1$.
- $\vec{\sigma} \models \mathsf{NZ}(\alpha)$ if and only if $\mathsf{Pr}_{\mathcal{C}_{\vec{\sigma}}}(\{\pi \in \mathrm{Paths}(\mathcal{C}_{\vec{\sigma}}, s^0) : \pi \models \alpha\}) > 0$.

Core

Definition (Deviation)

A deviation is a joint strategy $\vec{\sigma}_A$ for the coalition $A \subseteq \mathrm{N}$, with $A \neq \varnothing$.

Definition (Beneficial Deviation)

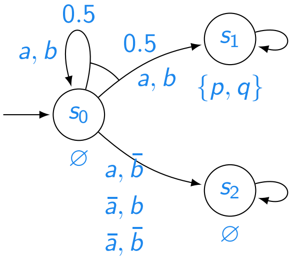
For a strategy profile $\vec{\sigma}$, we say $\vec{\sigma}'_A$ is a beneficial deviation from $\vec{\sigma}$ if $A \subseteq \mathrm{Lose}(\vec{\sigma})$ and for all $\vec{\sigma}'_{-A}$, we have $A \subseteq \mathrm{Win}((\vec{\sigma}'_A, \vec{\sigma}'_{-A}))$.

Definition (Core)

The core of a game $\mathcal{G}$, denoted $core(\mathcal{G})$, is then defined to be the set of strategy profiles that admit no beneficial deviation.

Example

Consider a game with two players $\mathsf{N} = \{1, 2\}$ and two variables $\mathsf{AP} = \{p, q\}$, with player 1's action set being $\mathsf{Ac}_1 = \{a, \bar{a}\}$ and player 2's being $\mathsf{Ac}_2 = \{b, \bar{b}\}$. Let $\gamma_1 = \mathbf{F} p$ and $\gamma_2 = \mathbf{F} q$.



consider a strategy profile $\vec{\sigma}$ in which player 1/2 always chooses action a/b in s_0 (*i.e.*, chooses a/b with probability 1)

consider a strategy profile $\vec{\sigma}'$ in which player 1/2 chooses action $\bar{a}/\bar{b}$ with non-zero probability

Non-Coop: NE-Membership

NE-Membership

Given: Game $\mathcal{G}$, strategy profile $\vec{\sigma}$.

Question: Is $\vec{\sigma}$ a Nash equilibrium in the game $\mathcal{G}$?

In general, infinite memory strategies are needed to play ω-regular games with almost-sure winning conditions. Here, we assume that $\vec{\sigma}$ can be represented by some finite state machine.

Non-Coop: NE-Membership

1. For $i \in \mathrm{N}$:
   1.1 If $\pi(\vec{\sigma}) \not\models \mathsf{AS}(\gamma_i)$ then
      1.1.1 If there is σ_i' s.t. $(\vec{\sigma}_{-i}, \sigma_i') \models \mathsf{AS}(\gamma_i)$ then return "NO"
2. Return "YES"

- 1.1 amounts to qualitative model checking LTL formula γ_i over a Markov chain (PSPACE)
- 1.1.1 amounts to qualitative model checking LTL formula γ_i over a MDP (2EXPTIME)
- Lower-bound: reduce to qualitative model checking LTL formula γ_i over a MDP

Theorem

NE-Membership for probabilistic systems is 2EXPTIME-complete

Non-Coop: E/A-Nash

E-Nash

Given: Game $\mathcal{G}$, temporal property φ.

Quest: Is there any Nash equilibrium $\vec{\sigma}$ in $\mathcal{G}$ such that $\pi(\vec{\sigma}) \models \mathsf{AS}(\varphi)$?

- Use the similar construction and NE characterisation to deterministic games
- Use Qualitative Parity Logic (QPL) Realizability problem to find NE run inside punishing region
- Procedure is 2EXPTIME, lower-bound via LTL model checking over MDPs.

Theorem

E-Nash and A-Nash for probabilistic systems are 2EXPTIME-complete

Coop: E/A-Core

E-Core

Given: Game $\mathcal{G}$, temporal property φ.

Quest: Is there any core $\vec{\sigma}$ in $\mathcal{G}$ such that $\vec{\sigma} \models \mathsf{AS}(\varphi)$?

- Turn the game into its corresponding parity game
- Check for each possible winning coalition $\mathsf{Win} \subseteq \mathsf{N}$ s.t. for each possible losing coalition $\mathsf{Lose} \subseteq \mathsf{N} \setminus \mathsf{Win}$, there is no beneficial deviation
- Use QPL to solve some problems in the procedure
- Procedure is 2EXPTIME, lower-bound via LTL model checking over MDPs.

Theorem

E-Core and A-Core for probabilistic systems are 2EXPTIME-complete

Coop: Core-Membership

Core-Membership

Given: Game $\mathcal{G}$, a strategy profile $\vec{\sigma}$.

Quest: Is $\vec{\sigma}$ a core in $\mathcal{G}$?[a]

[a] Again, we assume that $\vec{\sigma}$ can be represented by some FSM

- For each $\mathsf{Lose} \subseteq \mathsf{N} \setminus \mathsf{Win}(\vec{\sigma})$, check if there is beneficial deviation
- This amounts to model checking LTL formula over a MDP (2EXPTIME)
- Lower-bound: reduce to qualitative model checking LTL formula γ_i over a MDP

Theorem

Core-Membership for probabilistic systems is 2EXPTIME-complete

Results

Deterministic	Non-Coop.	Coop.
E-(Nash/Core)	2EXPTIME-C	2EXPTIME-C
A-(Nash/Core)	2EXPTIME-C	2EXPTIME-C
(NE/Core)-Mbrshp	PSPACE-C	2EXPTIME-C

Table 1: Complexity results for deterministic systems.

Probabilistic	Non-Coop.	Coop.
E-(Nash/Core)	2EXPTIME-C	2EXPTIME-C
A-(Nash/Core)	2EXPTIME-C	2EXPTIME-C
(NE/Core)-Mbrshp	2EXPTIME-C	2EXPTIME-C

Table 2: Complexity results for probablisitc systems.

Part II: Repairing

Dealing with missing or bad equilibria

Problem

Individually rational choices can cause outcomes that are highly undesirable, *e.g.*, there is no equilibrium or the temporal specification is not satisfied.

Question

The problem with this is intrinsic in the system. Can we repair it in order to gain (desirable) equilibria?

Solution

Equilibrium Design: redesign the game such that individually rational behaviour leads to desired outcomes.

Equilibrium Design via Subsidy Scheme

Subsidy scheme

Let $\mathcal{G} = (A, w_1, \ldots, w_n)$ be a Mean-payoff game.
A subsidy scheme for $\mathcal{G}$ is a function $\kappa : \mathsf{N} \times \mathsf{St} \to \mathbb{N}$.
The cost of κ is $\mathsf{cost}(\kappa) = \sum_{i \in \mathsf{N}} \sum_{s \in \mathsf{St}} \kappa(i)(s)$.

Subsidised game

For a Mean-payoff game $\mathcal{G} = (A, w_1, \ldots, w_n)$ and a subsidy scheme κ, the subsidised game $(\mathcal{G}, \kappa) = (A, w'_1, \ldots, w'_n)$ is obtained by updating every player's objective with $w'_i(s) = w_i + \kappa(i)(s)$, for every $s \in \mathsf{St}$.

Intuition

Designers can incentivise players to achieve outcomes that are desirable from the temporal specification point of view.

Equilibrium Design Implementation

Definition (Weak Implementation)

For a given game $\mathcal{G}$, a temporal specification φ and a budget $\beta \in \mathbb{N}$, find a subsidy scheme κ with $\mathsf{cost}(\kappa) \leq \beta$ such that $(\mathcal{G}, \kappa, \varphi)$ solves $\text{E-N}\textsc{ash}$ positively.

Definition (Strong Implementation)

For a given game $\mathcal{G}$, a temporal specification φ and a budget $\beta \in \mathbb{N}$, find a subsidy scheme κ with $\mathsf{cost}(\kappa) \leq \beta$ such that $(\mathcal{G}, \kappa, \varphi)$ solves $\text{A-N}\textsc{ash}$ positively.

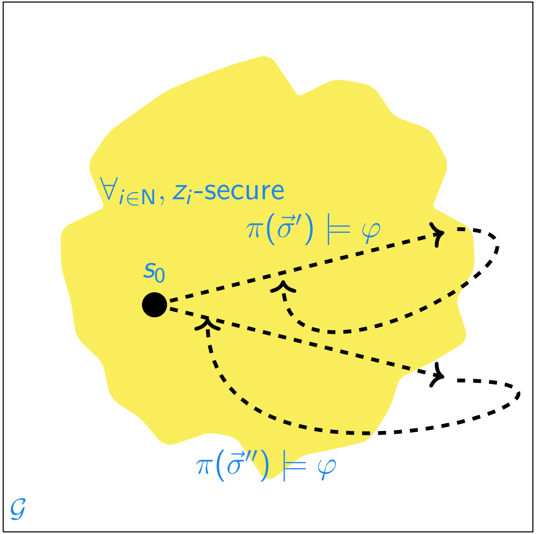
Theorem (Counting subsidy schemes)

The number of subsidy schemes with cost bounded by β *is*

$$|\mathcal{K}(\mathcal{G}, \beta)| = \frac{\beta + 1}{m} \cdot \binom{\beta + m}{\beta + 1}$$

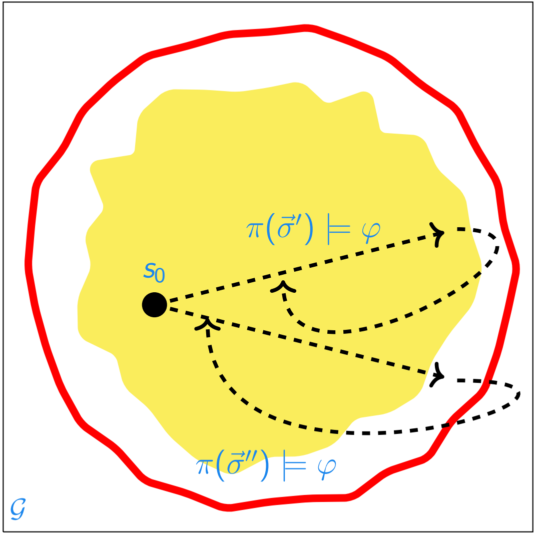
where $m = |St| \cdot |N|$

Solving Weak Implementation: Intuition



E-NASH:
$\exists \vec{\sigma} \in \mathrm{NE}(\mathcal{G}), \pi(\vec{\sigma}) \models \varphi$?
No :-(

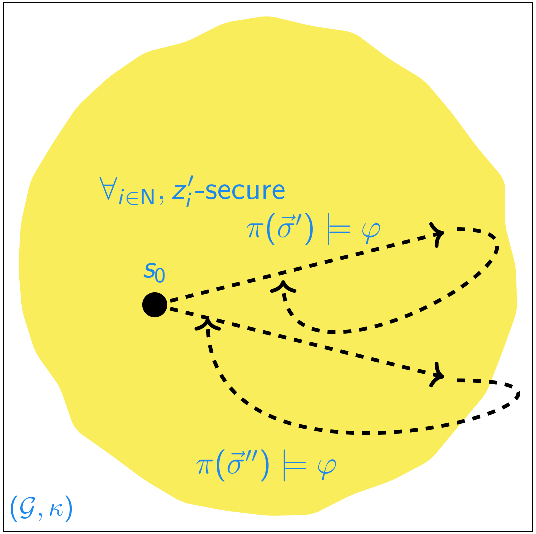
Solving Weak Implementation: Intuition



E-NASH:
$\exists \vec{\sigma} \in \mathsf{NE}(\mathcal{G}), \pi(\vec{\sigma}) \models \varphi$?
No :-(

If only we can modify the perimeter...

Solving Weak Implementation: Intuition



Apply subsidy scheme
$\kappa \in \mathcal{K}(\mathcal{G}, \beta)$

E-NASH:
$\exists \vec{\sigma} \in \mathsf{NE}(\mathcal{G}), \pi(\vec{\sigma}) \models \varphi$?
YES :-)

Complexity

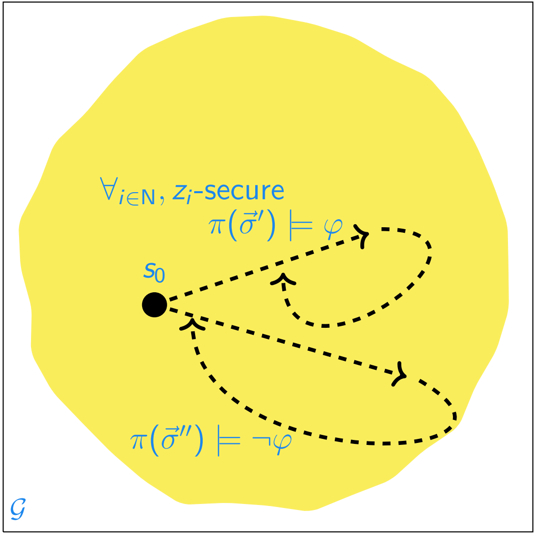
- For LTL specifications: PSPACE-complete (bottleneck is LTL model-checking)

Solving Weak Implementation: Complexity

Complexity

- For LTL specifications: PSPACE-complete (bottleneck is LTL model-checking)
- For GR(1) specifications: NP-complete (GR(1) model checking is poly, all guesses are made together)

Solving Strong Implementation: Intuition

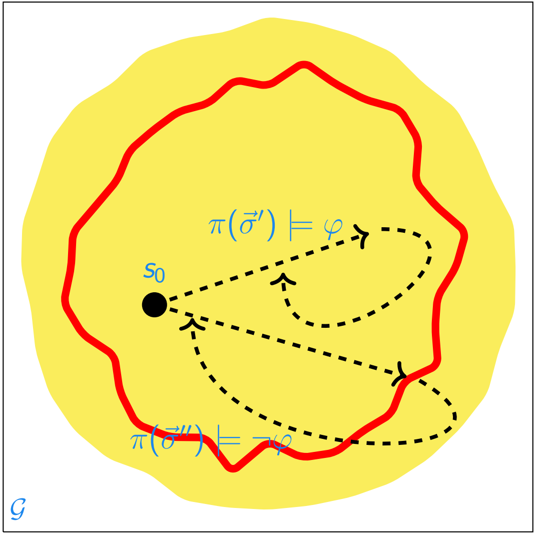


A-Nash:

$\forall \vec{\sigma} \in \mathrm{NE}(\mathcal{G}), \pi(\vec{\sigma}) \models \varphi$?

No :-(

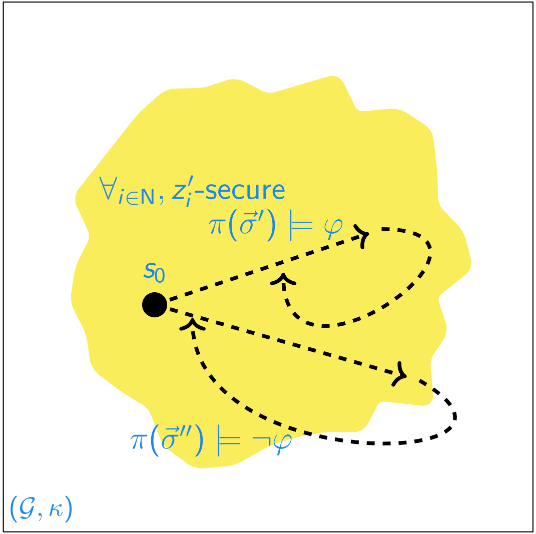
Solving Weak Implementation: Intuition



A-NASH:
$\forall \vec{\sigma} \in \mathsf{NE}(\mathcal{G}), \pi(\vec{\sigma}) \models \varphi$?
No :-(

If only we can modify the perimeter...

Solving Weak Implementation: Intuition



Apply subsidy scheme
$\kappa \in \mathcal{K}(\mathcal{G}, \beta)$

A-NASH:
$\forall \vec{\sigma} \in \mathsf{NE}(\mathcal{G}), \pi(\vec{\sigma}) \models \varphi$?
YES :-)

Solving Strong Implementation

Complexity

- For LTL specifications: PSPACE-complete (alternating quantification is absorbed)

Solving Strong Implementation

Complexity

- For LTL specifications: PSPACE-complete (alternating quantification is absorbed)
- For GR(1) specifications: Σ_2^P-complete (extra alternation is unavoidable)

Optimizing the budget

For a given game $\mathcal{G}$, we say that β is the optimal budget if it is the minimum required to solve weak or strong implementation, respectively.

Definition (Optimality)

Opt-WI For a game $\mathcal{G}$, compute the optimal budget β for the Weak Implementation.

Opt-SI For a game $\mathcal{G}$, compute the optimal budget β for the Strong Implementation.

Solving Optimality

Weak Implementation Complexity

- For LTL specifications: FPSPACE-complete (binary search is absorbed)
- For GR(1) specifications: $\mathrm{FP}^{\mathrm{NP}}$-complete. Hardness via TSP problem.

Strong Implementation Complexity

- For LTL specifications: PSPACE-complete (binary search is absorbed)
- For GR(1) specifications: $\mathrm{FP}^{\Sigma_2^P}$-complete. Hardness via Weighted MinQSAT$_2$ problem.

Checking the budget

Definition (Exactness)

Exact-WI For a game $\mathcal{G}$, check whether b is optimal for the Weak Implementation.

Exact-SI For a game $\mathcal{G}$, check whether b is the optimal for the Strong Implementation.

Checking uniqueness of the scheme

Definition (Uniqueness)

UOpt-WI For a game $\mathcal{G}$, check whether there is a unique subsidy scheme κ for the optimal budget β that solves the Weak Implementation.

UOpt-SI For a game $\mathcal{G}$, check whether there is a unique subsidy scheme κ for the optimal budget β that solves the Strong Implementation.

Complexity table summary

	LTL Spec.	GR(1) Spec.
Weak Implementation	PSPACE-complete	NP-complete
Strong Implementation	PSPACE-complete	Σ_2^P-complete
OPT-WI	FPSPACE-complete	FP^{NP}-complete
OPT-SI	FPSPACE-complete	$FP^{\Sigma_2^P}$-complete
EXACT-WI	PSPACE-complete	D^P-complete
EXACT-SI	PSPACE-complete	D_2^P-complete
UOPT-WI	PSPACE-complete	Δ_2^P-complete
UOPT-SI	PSPACE-complete	Δ_3^P-complete

Epilogue

- An approach to multi-agent systems correctness
- Decision problems and procedures to solve them
- A quest for tractable cases
- A different model with cooperative and probabilistic behaviour
- Future investigation: imperfect information, more quantitative flavour in probabilistic model, learning agents,...

Thank you!