Automated Temporal Equilibrium Analysis: Verification and Synthesis of Multi-Player Games

Julian Gutierrez[1] Muhammad Najib[2] Giuseppe Perelli[3] Michael Wooldridge[4]

30th International Joint Conference on Artificial Intelligence (IJCAI-21)

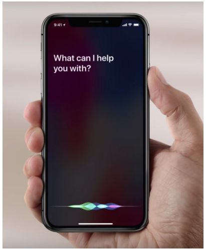
[1]Faculty of Information Technology, Monash University
[2]Department of Computer Science, University of Kaiserslautern
[3]Department of Computer, Automation, and Business Engineering, La Sapienza University of Rome
[4]Department of Computer Science, University of Oxford

AI Systems in Our Lives

- More AI systems integrated in our lives

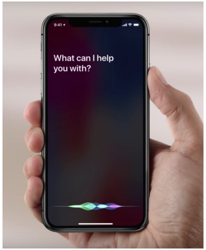
AI Systems in Our Lives

- More AI systems integrated in our lives
- E.g. Siri, Alexa, trading softwares, autonomous cars...

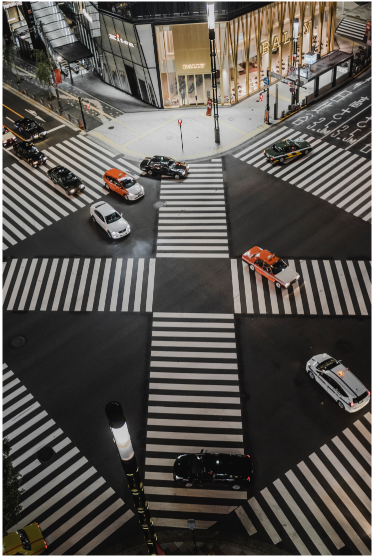


AI Systems in Our Lives

- More AI systems integrated in our lives
- E.g. Siri, Alexa, trading softwares, autonomous cars...
- Multiple interacting semi-autonomous software components (agents): multi-agent systems (MAS).

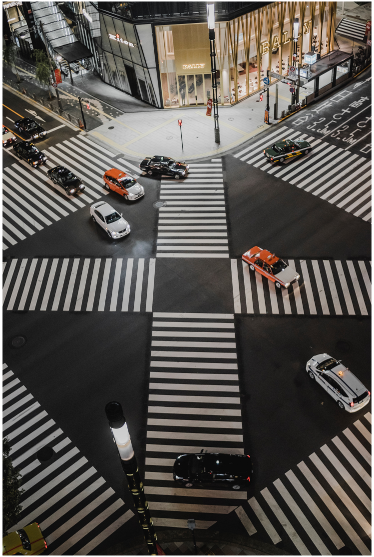


Multi-Agent Systems



- System composed of multiple entitites (players/agents): autonomous cars.

Multi-Agent Systems



- System composed of multiple entitites (players/agents): autonomous cars.
- Each agent may have different (not necessarily antagonistic) goal: each car has unique destination.

Multi-Agent Systems Correctness

How do we define correctness in MAS?

Multi-Agent Systems Correctness

How do we define correctness in MAS?

- We can use the “classical” approach in formal verification, however...

Multi-Agent Systems Correctness

How do we define correctness in MAS?

- We can use the "classical" approach in formal verification, however...
- Agents are rational

Multi-Agent Systems Correctness

How do we define correctness in MAS?

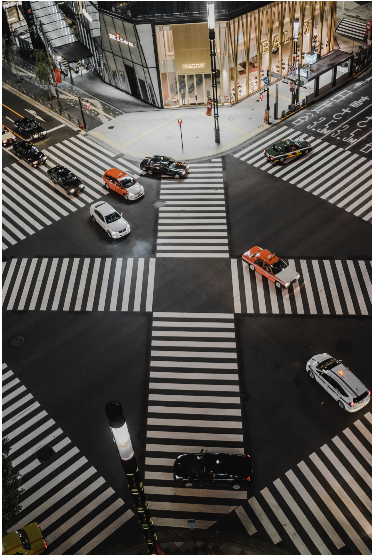
- We can use the "classical" approach in formal verification, however...
- Agents are rational
- Agents pursue their goals/preferences strategically

Multi-Agent Systems Correctness

How do we define correctness in MAS?

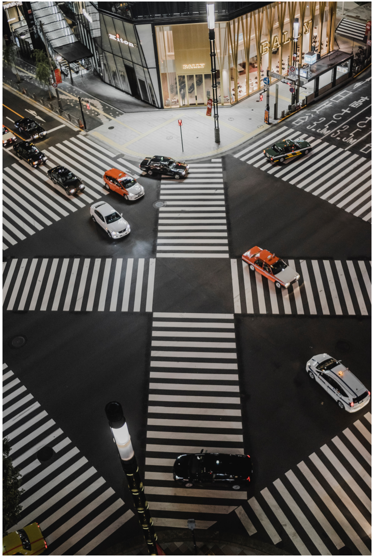
- We can use the "classical" approach in formal verification, however...
- Agents are rational
- Agents pursue their goals/preferences strategically
- Some *possible* behaviour may not arise

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



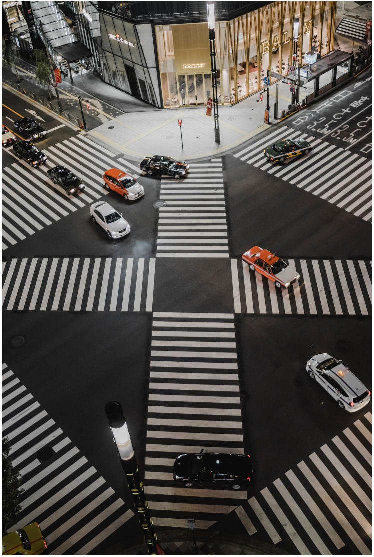
- Autonomous cars crossing an intersection

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



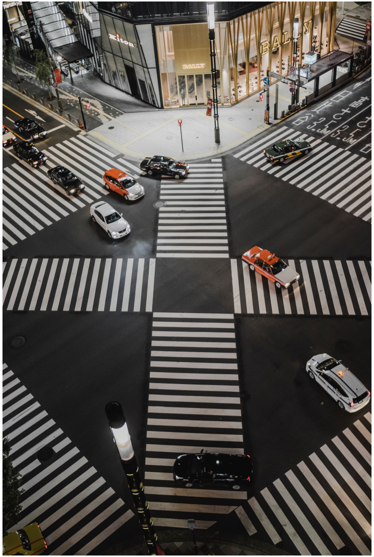
- Autonomous cars crossing an intersection
- Most of them (are expected to) cross without crashing with each other

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



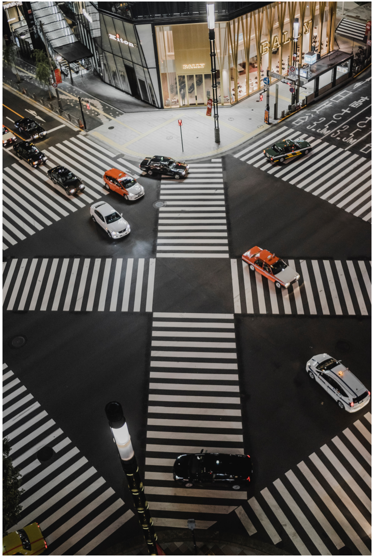
- Autonomous cars crossing an intersection
- Most of them (are expected to) cross without crashing with each other
- Cross and crash is also a *possible* behaviour of the system

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?

Multi-Agent Systems Correctness

How do we define correctness in MAS?

- Is the system correct with respect to the set of stable behaviours?

Multi-Agent Systems Correctness

How do we define correctness in MAS?

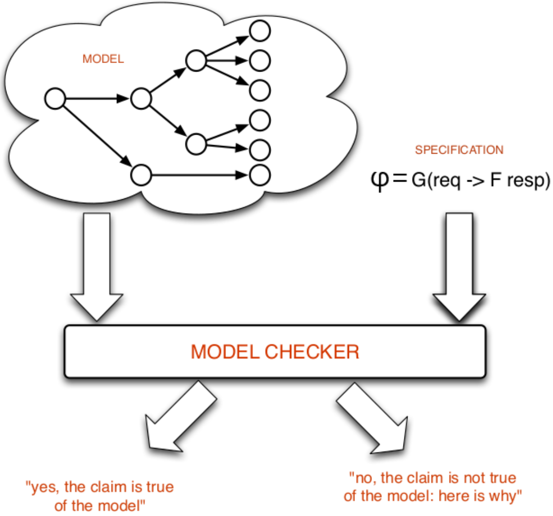
- Is the system correct with respect to the set of stable behaviours?
- Stable behaviours $\Rightarrow$ Nash equilibria via game theory

Multi-Agent Systems Correctness

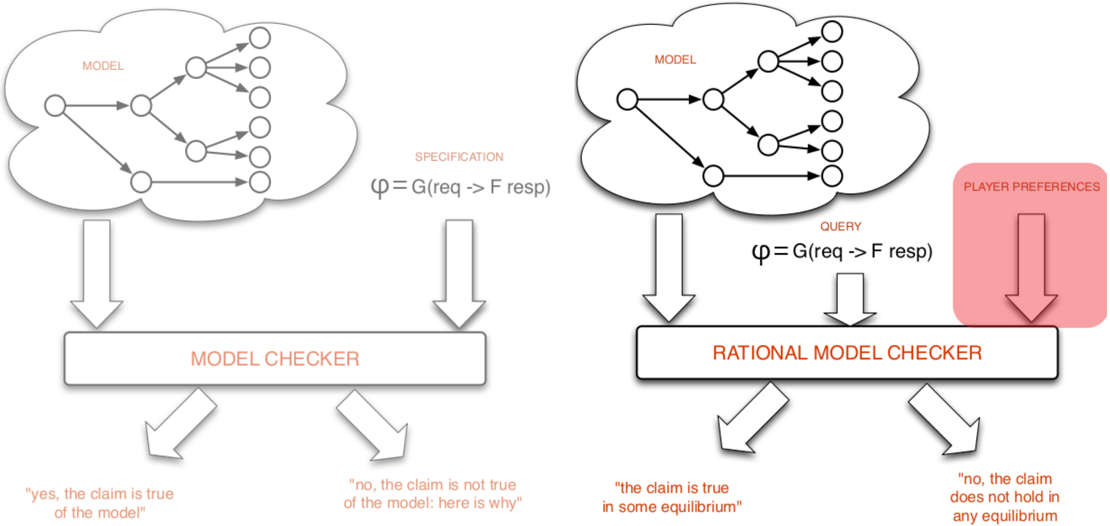
How do we define correctness in MAS?

- Is the system correct with respect to the set of stable behaviours?
- Stable behaviours $\Rightarrow$ Nash equilibria via game theory
- Turn MAS into multi-player game

From Verification to Rational Verification



From Verification to Rational Verification



E-Nash

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

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

Rational Verification

E-Nash

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

Question: 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.

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

Theorem (Complexity)

For the case of both the specification φ and the agents goals γ_i expressed as LTL formulas, rational verification is 2EXPTIME-Complete.[1]

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

Rational Verification

E-Nash

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

Question: 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.

Question: 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 $\mathcal{G}$.

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

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 structure (CGS) [2],
- γ_i is the LTL goal for player i.
- $\lambda : \mathsf{St} \to 2^{\mathsf{AP}}$ is a labelling function

LTL

$$\varphi ::= \top \mid p \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathbf{X}\varphi \mid \varphi \, \mathbf{U} \, \varphi$$

LTL formulae interpreted w.r.t. (π, t, λ), where π is a path over some multi-player game, $t \in \mathbb{N}$ is a temporal index into π.

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

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).

Games

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

- $\mathcal{M} = (\mathrm{N}, (\mathrm{Ac}_i)_{i \in \mathrm{N}}, \mathrm{St}, s_0, \mathrm{tr})$ is a concurrent game structure (CGS) [3],
- $\alpha_i : \mathrm{St} \to \mathbb{N}$ is the goal of player i, given as a priority function over St.

[3] 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.

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.

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 $\vec{\sigma} = \langle \sigma_1, \ldots, \sigma_N \rangle$ 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.

Nash Equilibria

Preference Relation

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 two strategy profiles $\vec{\sigma}$ and $\vec{\sigma}'$ in $\mathcal{G}$, we have

$$\pi(\vec{\sigma}) \succeq_i \pi(\vec{\sigma}') \text{ if and only if } \pi(\vec{\sigma}') \models w_i \text{ implies } \pi(\vec{\sigma}) \models w_i.$$

Nash Equilibria

Preference Relation

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 two strategy profiles $\vec{\sigma}$ and $\vec{\sigma}'$ in $\mathcal{G}$, we have

$$\pi(\vec{\sigma}) \succeq_i \pi(\vec{\sigma}') \text{ if and only if } \pi(\vec{\sigma}') \models w_i \text{ implies } \pi(\vec{\sigma}) \models w_i.$$

Nash Equilibrium

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

$$\pi(\vec{\sigma}) \succeq_i \pi((\vec{\sigma}_{-i}, \sigma'_i))$$

where $(\vec{\sigma}_{-i}, \sigma'_i)$ denotes $(\sigma_1, \ldots, \sigma_{i-1}, \sigma'_i, \sigma_{i+1}, \ldots, \sigma_n)$, the strategy profile where the strategy of player i in $\vec{\sigma}$ is replaced by σ'_i.
i.e., **no player can benefit by changing its strategy unilaterally.**

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 [4] *for every $j \in Lose(\pi)$.* [5] *Where $\vec{a}^k = \langle a_1, \ldots, a_n \rangle$ is an action profile at k.*

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

[4] Punishing secure: agent j does not have a strategy σ'_j that wins against $\vec{\sigma}_{-j}$, i.e. $\pi(\vec{\sigma}_{-j}, \sigma'_j) \models \gamma_j$.
[5] Here $\mathrm{Lose}(\pi) = \{j \in \mathrm{N} : \pi \not\models \gamma_j\}$ are the agents that are not satisfied over π.

NE Characterisation via Local Reasoning

- Memory is needed to satisfy LTL goal

NE Characterisation via Local Reasoning

- Memory is needed to satisfy LTL goal
- Memory is NOT necessary for (2-player) parity games (memoryless/positional determinacy)

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) $\mathcal{A}_i = \langle 2^{\mathsf{AP}}, Q, q^0, \rho, \alpha \rangle$.

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 $s'_0 = (s_0, q_1^0, \ldots, q_n^0)$;
  - 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

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 $\mathcal{G}_{PAR}$.*

Invariances

Lemma (Goal Invariance)

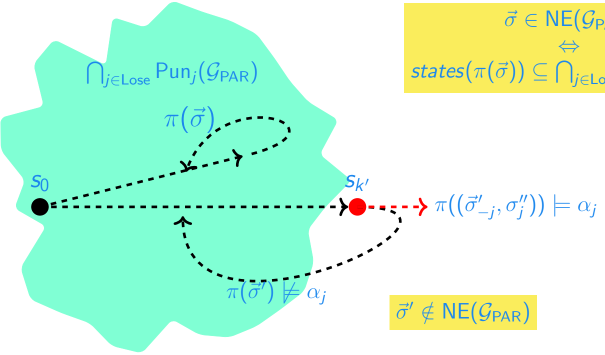
Let $\mathcal{G}_{\mathsf{LTL}}$ be an LTL game and $\mathcal{G}_{PAR}$ its associated Parity game. Then, for every strategy profile $\vec{\sigma}$ and player i, it is the case that $\pi(\vec{\sigma}) \models \gamma_i$ in $\mathcal{G}_{\mathsf{LTL}}$ if and only if $\pi(\vec{\sigma}) \models \alpha_i$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

Visualising NE Characterisation

$$\vec{\sigma} \in \mathsf{NE}(\mathcal{G}_{\mathsf{PAR}})$$
$$\Leftrightarrow$$
$$\mathit{states}(\pi(\vec{\sigma})) \subseteq \bigcap_{j \in \mathsf{Lose}} \mathsf{Pun}_j(\mathcal{G}_{\mathsf{PAR}})$$



$\vec{\sigma}' \notin \mathsf{NE}(\mathcal{G}_{\mathsf{PAR}})$

$\bigcap_{j \in \mathsf{Lose}} \mathsf{Pun}_j(\mathcal{G}_{\mathsf{PAR}})$ 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.

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

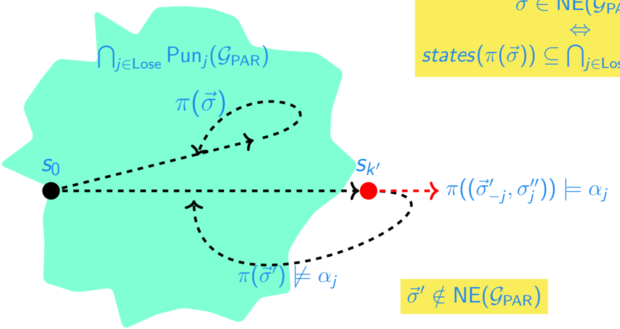
punishing region for $\mathsf{Lose} = \bigcap_{j \in \mathsf{Lose}} \mathsf{Pun}_j(\mathcal{G}_{\mathsf{PAR}})$

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

$$\vec{\sigma} \in \mathsf{NE}(\mathcal{G}_{\mathsf{PAR}})$$
$$\Leftrightarrow$$
$$\mathit{states}(\pi(\vec{\sigma})) \subseteq \bigcap_{j \in \mathsf{Lose}} \mathsf{Pun}_j(\mathcal{G}_{\mathsf{PAR}})$$



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 $\alpha_i, i \in \mathsf{Win} = \mathsf{N} \setminus \mathsf{Lose}$.

Finding NE Run

- $\pi(\vec{\sigma})$ must be accepting for each α_i, $i \in \mathsf{Win} = \mathsf{N} \setminus \mathsf{Lose}$.
- Solve emptiness problem of DPWs intersection $\bigtimes_{i \in \mathsf{Win}} \mathcal{A}^i$

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

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 $0 \leq i \leq m$

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 DSWs $\bigtimes_{i \in \mathsf{Win}} \mathcal{S}_i$ and nonemptiness check can be done in polynomial time

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"

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"

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"

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"

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"

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"

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.

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

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

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

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.

EVE (Equilibrium Verification Environment)

- Simple Reactive Modules Language (SRML)[6] as modelling language

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

EVE (Equilibrium Verification Environment)

- Simple Reactive Modules Language (SRML)[6] as modelling language
- Supports general-sum multi-player LTL games, bisimulation-invariant strategies, and perfect recall.

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

EVE (Equilibrium Verification Environment)

- Simple Reactive Modules Language (SRML)[6] as modelling language
- Supports general-sum multi-player LTL games, bisimulation-invariant strategies, and perfect recall.
- Supports Non-emptiness, E-Nash, and A-Nash

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

EVE (Equilibrium Verification Environment)

- Simple Reactive Modules Language (SRML)[6] 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

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

EVE (Equilibrium Verification Environment)

- Simple Reactive Modules Language (SRML)[6] 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/`

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

EVE vs Other Similar Tools

	EVE	**PRALINE**[7]	**MCMAS**[8]
Goal language	LTL	Büchi	LTL
Bisim. invariant strategies	Yes	No	No
Memoryful	Yes	Yes	No

[7] R. Brenguier. "PRALINE: A Tool for Computing Nash Equilibria in Concurrent Games". In: *CAV*. 2013.

[8] Petr Čermák et al. "MCMAS-SLK: A Model Checker for the Verification of Strategy Logic Specifications". In: *CAV*. 2014.

Non-Emptiness Experiment Result[9]

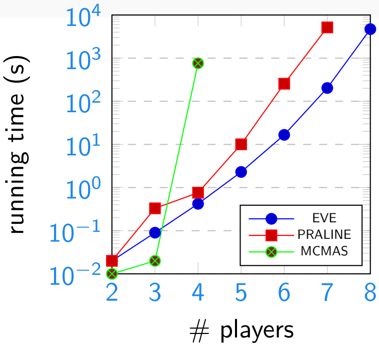


Figure 1: Running time for NON-EMPTINESS Gossip Protocol.

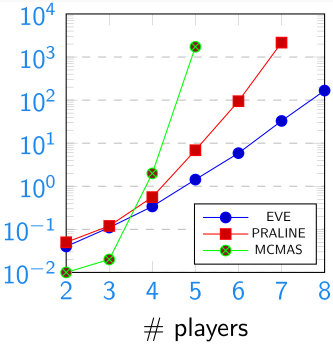


Figure 2: Running time for NON-EMPTINESS Replica Control Protocol.

Time-out was set to 7200 seconds (2 hours).

[9] Y-axis is in logarithmic scale.

E-Nash Experiment Result[10]

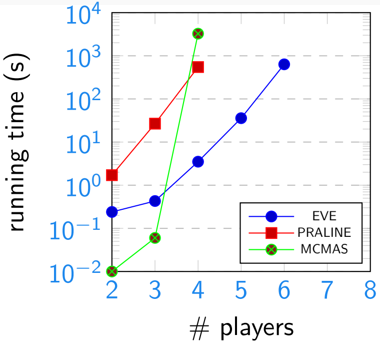


Figure 3: Running time for E-NASH Gossip Protocol.

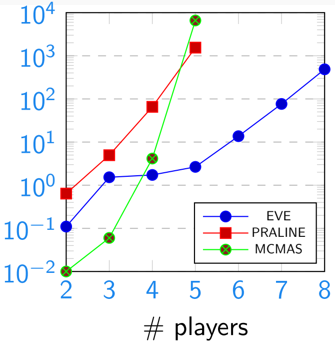


Figure 4: Running time for E-NASH Replica Control Protocol.

Time-out was set to 7200 seconds (2 hours).

[10] Y-axis is in logarithmic scale.

A-Nash Experiment Result[11]

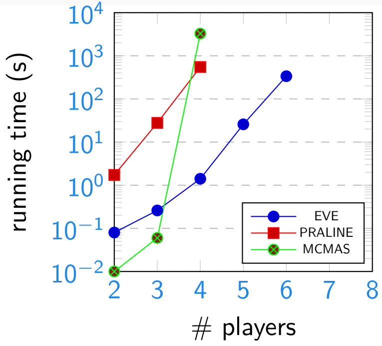


Figure 5: Running time for A-NASH Gossip Protocol.

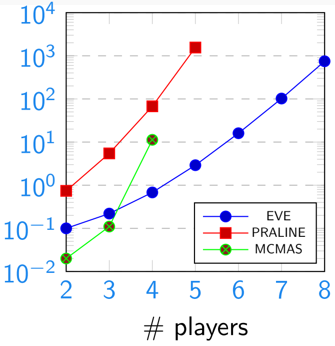


Figure 6: Running time for A-NASH Replica Control Protocol.

Time-out was set to 7200 seconds (2 hours).

[11] Y-axis is in logarithmic scale.

Conclusions

- Two main contributions:
  - Novel and optimal decision procedure for rational verification and synthesis
  - Complete and efficient implementation
- Future directions:
  - Cooperative setting: implementing "core"[12] as the solution concept
  - Probabilistic systems[13]
  - Decidable classes of imperfect information

[12] Julian Gutierrez, Sarit Kraus, and Michael Wooldridge. "Cooperative Concurrent Games". In: *AAMAS*. 2019.
[13] Julian Gutierrez et al. "Rational Verification for Probabilistic Systems". In: *KR*. to appear. 2021.