Verifying and Designing Equilibria in Multi-Agent Systems*

Muhammad Najib

Department of Computer Science, University of Oxford

RADICAL, Amsterdam, 26 August 2019

*Published works are joint work with Julian Gutierrez, Giuseppe Perelli, and Michael Wooldridge.

Overview

- Introduction
- Rational Verification via Parity Games
- Tractable Cases of Rational Verification
- Equilibrium Design in Concurrent Games
- Conclusions

Outline

- Introduction
- Rational Verification via Parity Games
- Tractable Cases of Rational Verification
- Equilibrium Design in Concurrent Games
- Conclusions

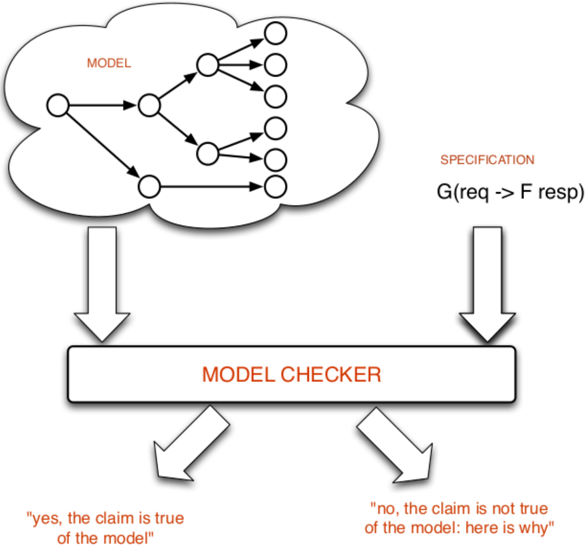
Correctness Problem

- How do we define correctness in multiagent systems?
- Each agent has her own goal, and the goals are not necessarily aligned
- Unlike classical verification, there is no single "litmus test" for system correctness

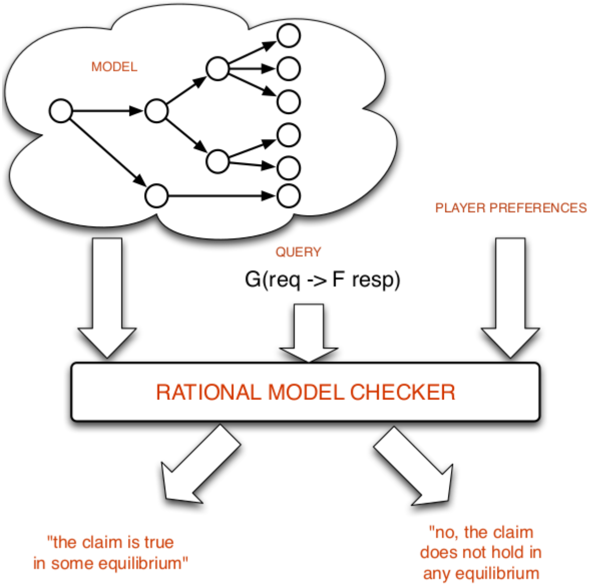
Correctness Problem

- Agents are rational
- Agents pursue their interests strategically
- An appropriate framework for studying strategic interaction between self-interested agents: **game theory**

(Classical) Model Checking



Equilibrium Checking



Arenas

Games are playing on graph-like arenas of the form:

$$A = \langle \mathrm{N}, \mathrm{Ac}, \mathrm{St}, s_0, \mathrm{tr}, \lambda \rangle$$

- N (finite) set of agents;
- Ac (finite) set of actions;
- St (finite) set of states (s_0 initial state);
- $\mathrm{tr} : \mathrm{St} \times \mathrm{Ac}^{\mathrm{N}} \to \mathrm{St}$ transition function [a];
- $\lambda : \mathrm{St} \to 2^{\mathrm{AP}}$ labelling function.

[a] At every state, agents take actions concurrently and move to the next state

Strategies

Strategy

Finite state machine $\sigma = \langle Q, \mathrm{St}, q_0, \delta, \tau \rangle$

- Q, internal state (q_0 initial state);
- $\delta : Q \times \mathrm{St} \to Q$ internal transition function;
- $\tau : Q \to \mathrm{Ac}$ action function.

A strategy is a recipe for the agent prescribing the action to take at every time-step of the game execution.

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.

Games and Nash Equilibria

A game $\mathcal{G} = \langle A, \mathsf{pay}_1, \ldots, \mathsf{pay}_{|N|} \rangle$ is defined by an arena and a list of payoff functions, one per each agent.
For a game $\mathcal{G}$, a strategy profile $\vec{\sigma}$ is a *Nash equilibrium* of $\mathcal{G}$ if, for every player i and strategy σ'_i, we have

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

i.e., **a player cannot improve her payoff by going "alone"**.

Rational Verification[†]

E-Nash

Given: a multiagent system $\mathcal{G}$ and a temporal logic formula φ.
Question: Is it the case that $\pi(\vec{\sigma}) \models \varphi$ in **some** $\vec{\sigma} \in NE(\mathcal{G})$?

Other rational verification problem:

- A-Nash: the dual of E-Nash (**all** $\vec{\sigma} \in NE(\mathcal{G})$)
- Non-Emptiness: special case of E-Nash ($\varphi = \top$)

[†] Michael Wooldridge et al. "Rational Verification: From Model Checking to Equilibrium Checking". In: *AAAI*. 2016.

Outline

- Introduction
- Rational Verification via Parity Games
- Tractable Cases of Rational Verification
- Equilibrium Design in Concurrent Games
- Conclusions

NE Characterisation

Lemma

π is sustained by a Nash equilibrium strategy profile iff every player j whose goal is not satisfied by π is punishable at π[a]

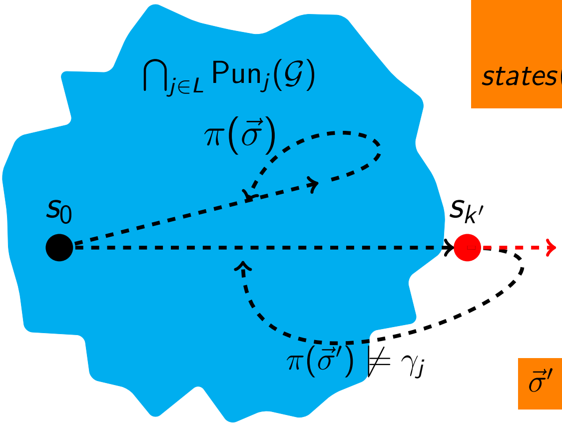
[a] Julian Gutierrez, Paul Harrenstein, and Michael Wooldridge. "Expresiveness and Complexity Results for Strategic Reasoning". In: *CONCUR*. 2015.

Nash equilibrium = Punishability + Memory

NE Characterisation

$L = \mathrm{N} \setminus W$

$\vec{\sigma} \in \mathrm{NE}(\mathcal{G})$
$\Leftrightarrow$
$\mathit{states}(\pi(\vec{\sigma})) \subseteq \bigcap_{j \in L} \mathrm{Pun}_j(\mathcal{G})$



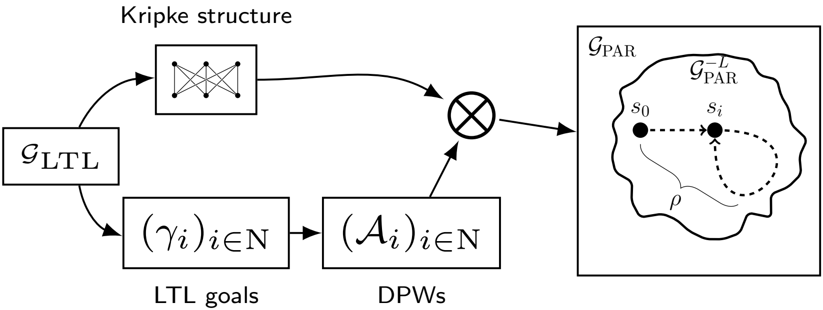
$\vec{\sigma}' \notin \mathrm{NE}(\mathcal{G})$

Why Parity?

- Memoryless/positional determinacy
- Solves the problem of keeping track deviating run
- Finite number of memoryless strategies
- Development of algorithms to solve PG (latest: quasipolynomial[‡])

[‡] Cristian S. Calude et al. "Deciding Parity Games in Quasipolynomial Time". In: *STOC*. 2017.

Workflow



Matches theoretical bound of 2EXPTIME for LTL Games

EVE (Equilibrium Verification Environment)

Open-source:

https://github.com/eve-mas/eve-parity

EVE Online: http://eve.cs.ox.ac.uk/eve

EVE vs Other Tools

	EVE	**PRALINE**	**MCMAS**
Goal language	LTL	Büchi	LTL
Bisim. invariant strategies	Yes	No	No
Memoryful	Yes	Yes	No

Non-Emptiness Experiment Result[§][¶]

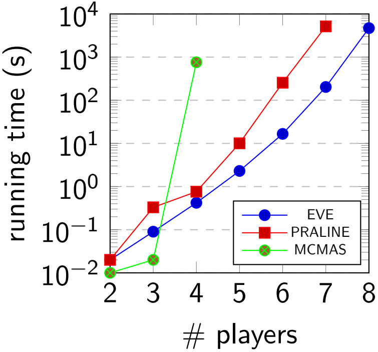


Figure 1: Running time for Non-Emptiness Gossip Protocol.

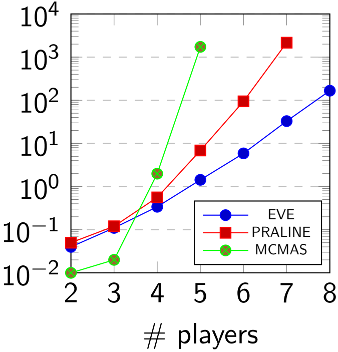


Figure 2: Running time for Non-Emptiness Replica Control Protocol.

[§] Y-axis is in logarithmic scale. Time-out was set to 7200 seconds (2 hours).

[¶] Julian Gutierrez et al. "EVE: A Tool for Temporal Equilibrium Analysis". In: *ATVA*. 2018.

Outline

- Introduction
- Rational Verification via Parity Games
- Tractable Cases of Rational Verification
- Equilibrium Design in Concurrent Games
- Conclusions

From Julian Gutierrez et al. “On Computational Tractability for Rational Verification”. In: *IJCAI*. 2019

GR(1)

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

$$(\mathbf{GF}\psi_1 \wedge \ldots \wedge \mathbf{GF}\psi_m) \rightarrow (\mathbf{GF}\phi_1 \wedge \ldots \wedge \mathbf{GF}\phi_n),$$

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

[‖] Roderick Bloem et al. "Synthesis of Reactive(1) designs". In: *Journal of Computer and System Sciences* (2012).

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, that is,

$$\mathrm{mp}(\beta) = \liminf_{n \to \infty} \mathrm{avg}_n(\beta)$$

where, for $n \in \mathbb{N}$, we define $\mathrm{avg}_n(\beta) = \frac{1}{n} \sum_{j=0}^{n-1} \beta_j$.

Games

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

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

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

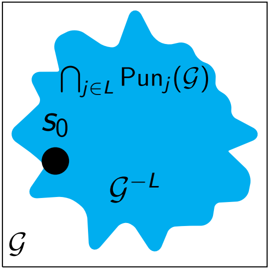
- $A = \langle \mathsf{N}, \mathsf{Ac}, \mathsf{St}, s_0, \mathsf{tr}, \lambda \rangle$ is an arena
- $\mathsf{w}_i : \mathsf{St} \to \mathbb{Z}$ is a function mapping every state of the arena into an integer number.

Cases

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

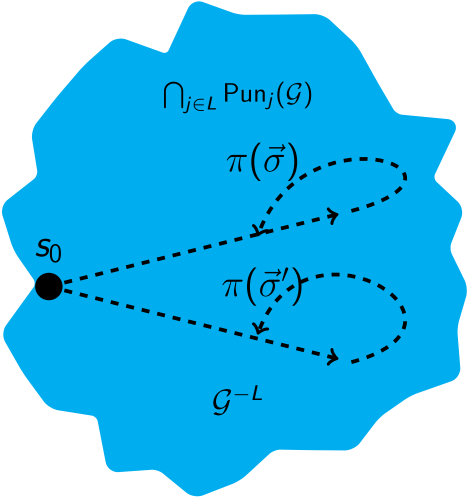
E-NASH in GR(1) games: the procedure

1. Obtain $\mathcal{G}^{-L}$ by computing punishment region $\mathrm{Pun}_j(\mathcal{G})$. Can be done in polynomial time with respect to the size of both $\mathcal{G}$ and γ_j via reduction to Streett game.



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

2. Check whether there exists an ultimately periodic path π in $\mathcal{G}^{-L}$ such that $\pi \models \varphi \wedge \bigwedge_{i \in W} \gamma_i$ holds.



$\pi(\vec{\sigma}) \overset{?}{\models} \varphi \wedge \bigwedge_{i \in W} \gamma_i$

$\pi(\vec{\sigma}') \overset{?}{\models} \varphi \wedge \bigwedge_{i \in W} \gamma_i$

E-NASH in GR(1) games with LTL spec.

Corollary (games with LTL specification)

The E-NASH *problem for GR(1) games with an LTL specification is PSPACE-complete.*

- Bottleneck: model checking LTL specification φ is PSPACE-complete.

E-Nash in GR(1) games with GR(1) spec.

Theorem (games with GR(1) specification)

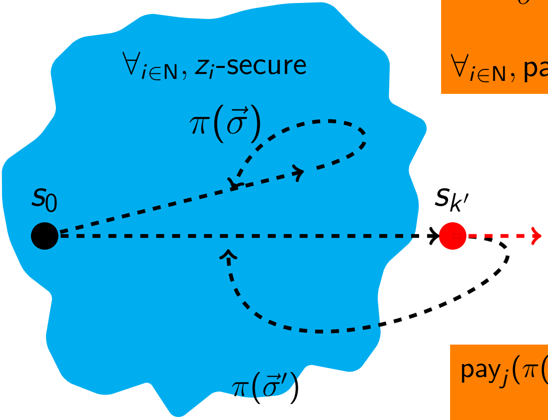
Can be solved in time that is polynomial in $|St|$, $|Ac|$, and $|\varphi|$, $|\gamma_1|, \ldots, |\gamma_N|$ and exponential in the number of players $|N|$.

- Streett automaton emptiness: can be solved in polynomial time w.r.t the automaton's index and its number of states and transitions**.
- The problem is fixed-parameter tractable (FPT), parameterised in the number of players.

**Monika Rauch Henzinger and Jan Telle. "Faster Algorithms for the Nonemptiness of Streett Automata and for Communication Protocol Pruning". In: *SWAT*. 1996, Orna Kupferman. "Automata Theory and Model Checking". In: *Handbook of TCS* (2015).

E-NASH in mp games: NE characterisation

$L = \mathrm{N} \setminus W$

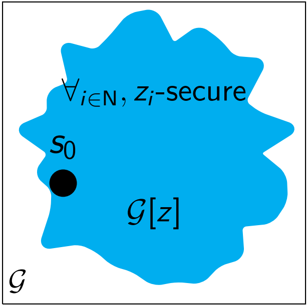


$$\vec{\sigma} \in \mathrm{NE}(\mathcal{G}) \Leftrightarrow \forall_{i \in \mathrm{N}}, \mathrm{pay}_i(\pi(\vec{\sigma})) \geq z_i$$

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

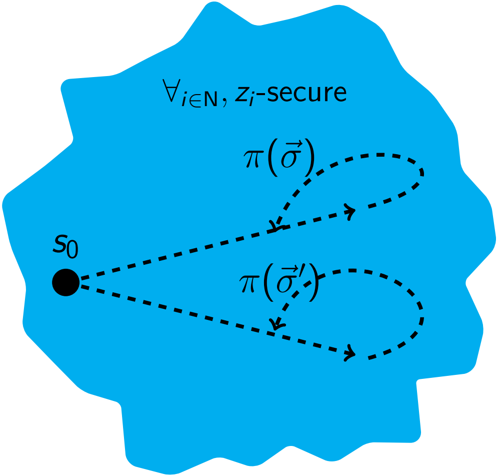
E-NASH in mp games: the procedure

1. Obtain $\mathcal{G}[z]$ by guessing vector $z \in \mathbb{R}^{\mathrm{N}}$ and remove "non-secure" states.



E-NASH in mp games: the procedure

2. Find an ultimately periodic path π in game $\mathcal{G}[z]$ such that $\pi \models \varphi$ and $z_i \leq \text{pay}_i(\pi)$ for every player $i \in \text{N}$.



$\pi(\vec{\sigma}) \overset{?}{\models} \varphi$

$\pi(\vec{\sigma}') \overset{?}{\models} \varphi$

E-Nash in mp games with LTL spec.

Corollary (mp games with LTL specification)

The E-Nash problem for mp games with an LTL specification formula φ is PSPACE-complete.

- Using $\mathsf{LTL}^{\mathsf{Lim}}$ model checking to find satisying run (PSPACE-complete[††]).

[††] Udi Boker et al. "Temporal Specifications with Accumulative Values". In: *ACM Transactions on Computational Logic* (2014).

E-NASH in mp games with GR(1) spec.

Theorem (mp games with GR(1) specification)

The E-NASH *problem for* mp *games with a GR(1) specification* φ *is NP-complete.*

- Define a linear program of size polynomial in $\mathcal{G}$ to find an ultimately-periodic run π satisfying GR(1) specification φ s.t. $\forall_{i \in \mathrm{N}}, z_i \leq \mathrm{pay}_i(\pi)$.
- Lower bound: with $\varphi = \top \Rightarrow$ NE existence in mp games[‡‡].

[‡‡] Michael Ummels and Dominik Wojtczak. "The Complexity of Nash Equilibria in Limit-Average Games". In: *CONCUR*. 2011.

Complexity Results

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

- NON-EMPTINESS:
  - LTL games: 2EXPTIME-complete
  - GR(1) games: PSPACE-complete/FPT
  - mp games: PSPACE-complete/NP-complete

Outline

- Introduction
- Rational Verification via Parity Games
- Tractable Cases of Rational Verification
- Equilibrium Design in Concurrent Games
- Conclusions

From Julian Gutierrez et al. “Equilibrium Design for Concurrent Games”. In: *CONCUR*. 2019

Designing Equilibrium

Equilibrium Design

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

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 $cost(\kappa) \leq \beta$ such that $(\mathcal{G}, \kappa, \varphi)$ solves E-Nash 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 $cost(\kappa) \leq \beta$ such that $(\mathcal{G}, \kappa, \varphi)$ solves A-Nash positively.

Optimising 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.

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

Outline

- Introduction
- Rational Verification via Parity Games
- Tractable Cases of Rational Verification
- Equilibrium Design in Concurrent Games
- Conclusions

Conclusions & Future Work

- Developed & implemented algorithmic techniques for rational verification
- Identified tractable cases for rational verification
- Introduced the concept of equilibrium design, and analysed complexity

Future work:

- Cooperative games
- Decidable cases of imperfect information games
- Consider social welfare in designing equilibrium