Game-Theoretic Verification of Multi-Agent Systems[1]

Part I: Introduction

Muhammad Najib
Heriot-Watt University

The 24th European Agent Systems Summer School
(EASSS 2024)

[1] Adapted from lecture slides by Mike Wooldridge (`mjw@cs.ox.ac.uk`) and Julian Gutierrez (`Julian.Gutierrez@monash.edu`).

Overview

1. Intro: verification, LTL, Games
2. Logic and games: Boolean games, adding cost
3. LTL and games: iterated Boolean games
4. Reactive modules games: rational verification in RMGs, EVE, complexity proof

Multi-Agent Systems Finally Happens!

Multi-agent systems today

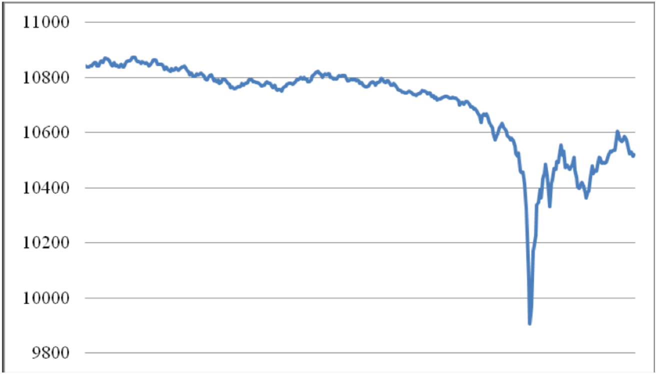
- Thirty years after it was first proposed, agent paradigm is now mainstream: Siri, Alexa, Cortana…
- Next: Siri talking to Siri — **multi-agent systems**
- But multi-agent systems are already used today
- High frequency ("algorithmic") traders are exactly that

Unpredictable Dynamics

- Unfortunately, multi-agent systems are prone to instability and have unpredictable dynamics
- October 1987 Market Crash:
  - the “big bang” led to automated trading systems for first time
  - simple feedback loops contributed to collapse in market
- May 2010 Flash Crash:
  - over a 30 minute period, Dow Jones lost over a trillion dollars
  - Accenture briefly traded at a penny a share
  - markets swiftly recovered (ish)

The Flash Crash

Dow Jones Industrial Average, 6 May 2010

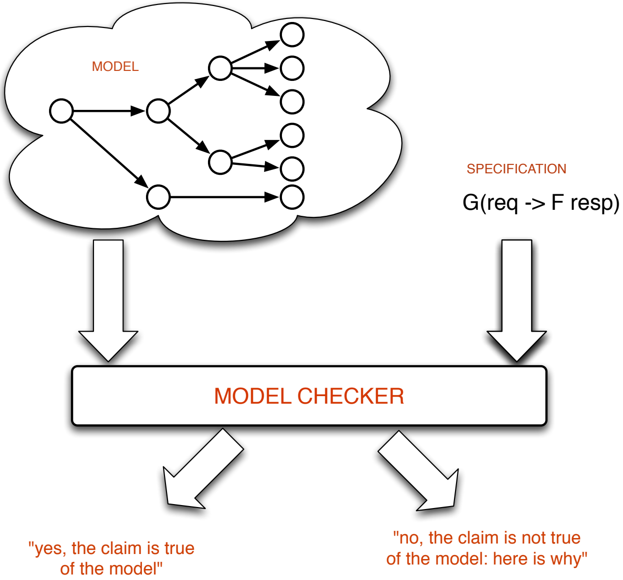


The Research Challenge

- Understanding and managing multi-agent dynamics is **essential**.
- Treat the flash crash as a **bug** and try to understand it using ideas from **verification** and **game theory**.

- Most successful approach to correctness.
- Idea is to view the state transition graph of a program P as a model M_P for temporal logic, and express correctness criteria as formula φ of temporal logic
- Verification then reduces to a model checking problem: $M_P \models \varphi$
- Hugely successful technique, widely used (SPIN, SMV, PRISM, MOCHA, MCMAS…)
- Most widely used logical specification language: LTL.

Model Checking



Linear Temporal Logic (LTL)

A standard language for talking about **infinite state sequences**.

$\top$	truth constant
p	primitive propositions ($\in \Phi$)
$\neg\varphi$	classical negation
$\varphi \vee \psi$	classical disjunction
$\mathbf{X}\varphi$	in the next state…
$\mathbf{F}\varphi$	will eventually be the case that φ
$\mathbf{G}\varphi$	is always the case that φ
$\varphi\,\mathbf{U}\,\psi$	φ until ψ

Example LTL formulae

$$\mathbf{F}\neg \mathit{jetlag}$$

eventually I will not have jetlag (a **liveness** property)

Example LTL formulae

$$\mathbf{G}\neg \mathit{crash}$$

the plane will never crash (a **safety** property)

Example LTL formulae

$\mathbf{GF}\,\mathit{drinkBeer}$

Example LTL formulae

$\mathbf{GF}\mathit{drinkBeer}$

I will drink beer **infinitely often**

Example LTL formulae

$\mathbf{FG}\mathit{dead}$

Example LTL formulae

$\mathbf{FG}\,\mathit{dead}$

Eventually will come a time at which I am dead forever after.

Example LTL formulae

$$(\neg \mathit{friends})\ \mathbf{U}\ \mathit{youApologise}$$

Example LTL formulae

$$(\neg \textit{friends})\ \mathbf{U}\ \textit{youApologise}$$

we are not friends **until** you apologise

LTL Model Checking

- Complexity of LTL model checking: PSPACE-complete
  Assumes state transition graph is **explicitly represented** in the input.
- Basic model checking questions:
  - **reachability**: is there some computation of the system on which φ eventually holds?
  - **invariance**: does φ hold on all computations of the system?

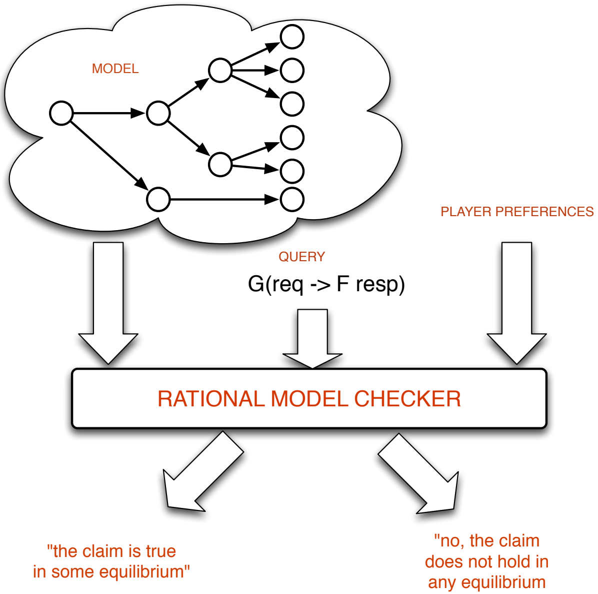
Assumptions in the Classical View of Correctness

- The standard model of verification assumes an **absolute standard of correctness**
- The specifier is able to say “the system is correct” or “the system is not correct”
- The specifier enjoys a **privileged position**
- For many systems, this is simply not appropriate. . .
- It makes no sense to ask whether the internet is “correct”!
- So what can we do instead?

Rational Verification

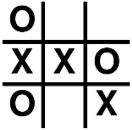
- We adopt a **game theoretic** standpoint
- Assume system components are **rational actors**, and that they act as best they can to bring about their **preferences**
- Appropriate analytical concepts are then **game theoretic solution concepts**, in particular, **equilibrium properties** such as **Nash equilibrium**
- Reachability and invariance are not appropriate in this setting: we are interested in whether properties will obtain **under the assumption of rational action**
- Some computations will not arise because they involve irrational action
- Key concepts:
  "Nash reachability" (**E-Nash**) and
  "Nash invariance" (**A-Nash**)

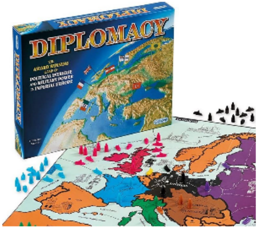
Rational Verification



Games

- Game theoretic standpoint: turn MAS into games
- What is a game?






What is a Game?

Ingredients:

1. Several decision makers: **players** or **agents**
2. Players have different **goals**
3. Each player can **act** to affect the outcome

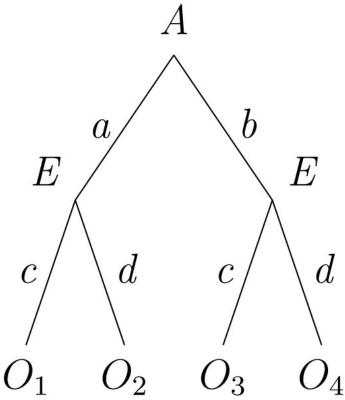
Types of Games

Two major types (in Economics):

1. Extensive form
2. Strategic/Normal form

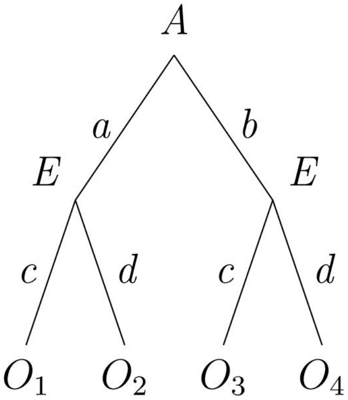
Extensive Form Games

- Explicit temporal structure



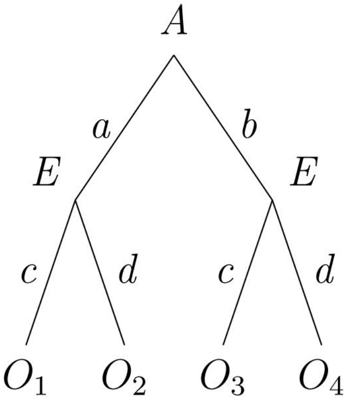
Extensive Form Games

- Explicit temporal structure
- Each non-terminal node owned by one player (whose turn)



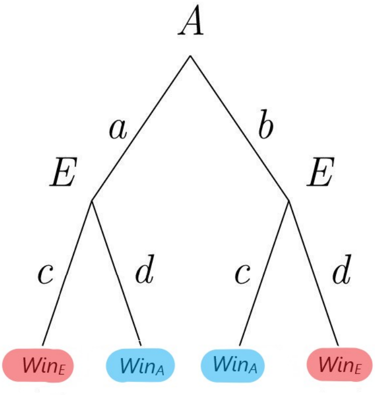
Extensive Form Games

- Explicit temporal structure
- Each non-terminal node owned by one player (whose turn)
- Edges correspond to possible actions



Extensive Form Games

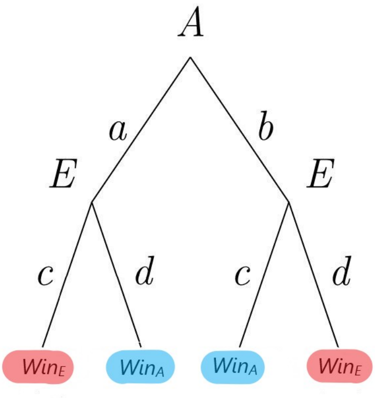
Is there a NE?



NE: a strategy profile where no player could benefit by changing their own strategy (holding all other players' strategies fixed)

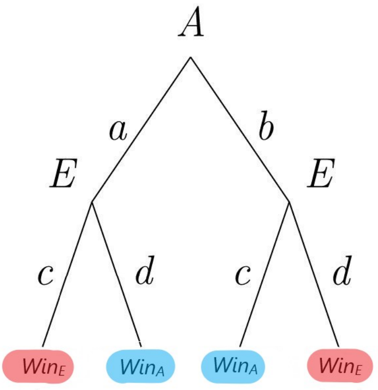
Extensive Form Games

Who has a winning strategy? **A**belard or **E**loise?



Extensive Form Games

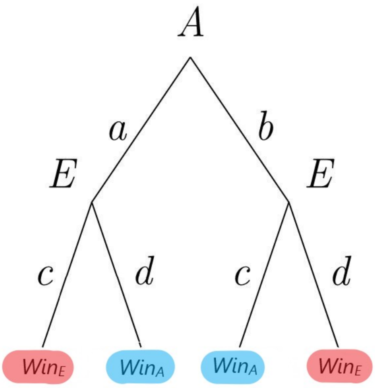
Who has a winning strategy? **A**belard or **E**loise? **Eloise.**



Extensive Form Games

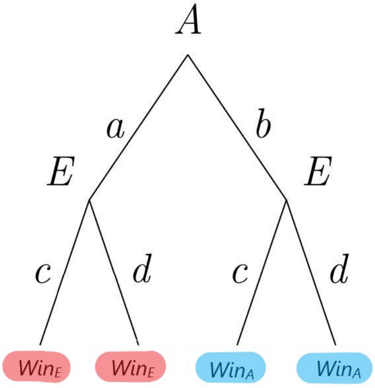
Who has a winning strategy? **A**belard or **E**loise? **Eloise**.

If Abelard chooses *a* then choose *d*, else choose *c*.



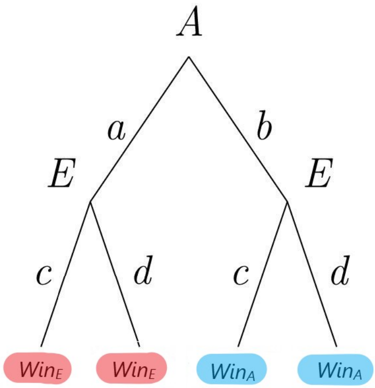
Extensive Form Games

What about this?



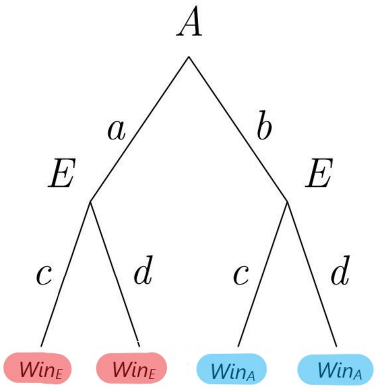
Extensive Form Games

What about this? **Abelard.**



Extensive Form Games

What about this? **Abelard.**
Choose *b*.



Strategic/Normal Form Games

- Emphasise players' **available strategies**

$$
\begin{array}{cc|c|c|}
 & & \multicolumn{2}{c}{E} \\
\cline{3-4}
 & & a & b \\
\cline{2-4}
\multirow{2}{*}{A} & \multicolumn{1}{|c|}{c} & O_1 & O_2 \\
\cline{2-4}
 & \multicolumn{1}{|c|}{d} & O_3 & O_4 \\
\cline{2-4}
\end{array}
$$

Strategic/Normal Form Games

- Emphasise players' **available strategies**
- No temporal structure

		E	
		a	b
A	c	O_1	O_2
	d	O_3	O_4

Strategic/Normal Form Games

Is there a NE?

		E	
		a	b
A	c	Win_A	Win_E
	d	Win_A	Win_E

Strategic/Normal Form Games

Is there a NE? Who has winning strategy?

		E	
		a	b
A	c	Win_A	Win_E
	d	Win_A	Win_E

Is there a NE? Who has winning strategy? **Eloise.**

		E	
		a	b
A	c	Win_A	Win_E
	d	Win_A	Win_E

Strategic/Normal Form Games

Is there a NE? Who has winning strategy? **Eloise. Choose *b***

		E	
		a	b
A	c	Win_A	Win_E
	d	Win_A	Win_E

Strategic/Normal Form Games

Who has winning strategy?

		E	
		a	b
A	c	Win_A	Win_E
	d	Win_E	Win_A

Strategic/Normal Form Games

Who has winning strategy? **Nobody.**

		E: a	E: b
A	c	Win_A	Win_E
	d	Win_E	Win_A

Strategic/Normal Form Games

Who has winning strategy? **Nobody.**

E

		a	b
A	c	Win_A	Win_E
	d	Win_E	Win_A

No NE...

Which model is appropriate for the Rock-Paper-Scissors game?

Which model is appropriate for the Rock-Paper-Scissors game?

Rock, Paper, Scissors	Rock	Paper	Scissors
Rock	0, 0	-1, 1	1, -1
Paper	1, -1	0, 0	-1, 1
Scissors	-1, 1	1, -1	0, 0

Figure: From http://gametheory101.com/

Which model is appropriate for the Rock-Paper-Scissors game?

Rock, Paper, Scissors	Rock	Paper	Scissors
Rock	0, 0	-1, 1	1, -1
Paper	1, -1	0, 0	-1, 1
Scissors	-1, 1	1, -1	0, 0

Figure: From http://gametheory101.com/

Is there a NE?

Game-Theoretic Verification of Multi-Agent Systems[2]

Part II: Logic and Games

Muhammad Najib
Heriot-Watt University

The 24th European Agent Systems Summer School
(EASSS 2024)

[2] Adapted from lecture slides by Mike Wooldridge (`mjw@cs.ox.ac.uk`) and Julian Gutierrez (`Julian.Gutierrez@monash.edu`).

Boolean Games

Boolean Games

- A natural class of **compactly specified** games
- Important from point of view of logic, games, multi-agent systems
- Basic idea is to specify player preferences via **logical formula**.
- Players strictly prefer to get their goal achieved rather than otherwise.

Reminder: Normal Form Games

A normal form game is given by a structure

$$G = (N, \Sigma_1, \ldots, \Sigma_n, u_1, \ldots, u_n)$$

where:

- $N = \{1, \ldots, n\}$ is the set of players
- Σ_i is the set of **strategies** (**choices**) for $i \in N$;
- $u_i : \Sigma_1 \times \cdots \times \Sigma_n \to \mathbb{R}$ is the **utility function** for i, which captures i's preferences.

Each player i must choose an element of Σ_i. When players have made choices, the resulting **strategy profile** $\vec{\sigma} = (\sigma_1, \ldots, \sigma_n)$ gives player i utility $u_i(\sigma_1, \ldots, \sigma_n)$. Players aim to **maximise utility**.

Reminder: Nash Equilibrium

- A collection of choices $(\sigma_1, \ldots, \sigma_n)$ is an NE if no player could benefit by unilaterally deviating.
- This means there is no player i and choice $\sigma_i' \in \Sigma_i$ such that

$$u_i(\sigma_1 \ldots, \sigma_i', \ldots, \sigma_n) > u_i(\sigma_1 \ldots, \sigma_i, \ldots, \sigma_n).$$

- NE is the basic concept of rational choice in normal form games.

Boolean Games

Formally, a Boolean game G is given by:

Boolean Games

Formally, a Boolean game G is given by:

- $N = \{1, \ldots, n\}$
  the **players**

Boolean Games

Formally, a Boolean game G is given by:

- $N = \{1, \ldots, n\}$
  the **players**
- $\Phi = \{p, q, \ldots\}$
  a finite set of **Boolean variables**

Boolean Games

Formally, a Boolean game G is given by:

- $N = \{1, \ldots, n\}$
  the **players**
- $\Phi = \{p, q, \ldots\}$
  a finite set of **Boolean variables**
- $\Phi_i \subseteq \Phi$ for each $i \in N$
  the **set of variables under the control of** i: we require:
  - $\Phi_i \cap \Phi_j = \emptyset$ for $i \neq j$
  - $\Phi_1 \cup \cdots \cup \Phi_n = \Phi$.

The assignments that i can make to Φ_i are the **actions/strategies** available to i.

Boolean Games

Formally, a Boolean game G is given by:

- $N = \{1, \ldots, n\}$
  the **players**
- $\Phi = \{p, q, \ldots\}$
  a finite set of **Boolean variables**
- $\Phi_i \subseteq \Phi$ for each $i \in N$
  the **set of variables under the control of** i: we require:
  - $\Phi_i \cap \Phi_j = \emptyset$ for $i \neq j$
  - $\Phi_1 \cup \cdots \cup \Phi_n = \Phi$.

  The assignments that i can make to Φ_i are the **actions/strategies** available to i.
- γ_i for each $i \in N$
  goal of agent i – the **specification** for i – propositional logic formula over Φ.

Outcomes

- A **strategy** for agent *i* is an assignment

$$\sigma_i : \Phi_i \to \mathbb{B}$$

  Agent *i* chooses a value for all its variables.
- An **strategy profile** is a **collection of choices**, one for each agent:

$$\vec{\sigma} = (\sigma_1, \ldots, \sigma_n)$$

- A strategy profile induces a propositional valuation: we write

$$\vec{\sigma} \models \varphi$$

  to mean that φ is satisfied by the valuation induced by $\vec{\sigma}$.
- A strategy profile will thus either satisfy/fail to satisfy each player's goal.

Utilities

For each player i we can define a utility function over strategy profiles — player gets utility 1 if goal satisfied, 0 otherwise:

$$u_i(\vec{\sigma}) = \begin{cases} 1 & \text{if } \vec{\sigma} \models \gamma_i \\ 0 & \text{otherwise.} \end{cases}$$

Preferences:

- Players strictly prefer to get their goal achieved than otherwise.
- Indifferent between outcomes that satisfy goal.
- Indifferent between outcomes that fail to satisfy goal.

A Boolean game thus induces a **normal form game**.

An Example

Suppose:

$$
\begin{aligned}
\Phi_1 &= \{p\} \\
\Phi_2 &= \{q, r\} \\
\gamma_1 &= q \\
\gamma_2 &= q \vee r
\end{aligned}
$$

What are the NE?

Another Example

$$
\begin{aligned}
\Phi_1 &= \{p\} \\
\Phi_2 &= \{q\} \\
\gamma_1 &= p \leftrightarrow q \\
\gamma_2 &= \neg(p \leftrightarrow q)
\end{aligned}
$$

What are the NE?

Another Example

$$\begin{aligned}\Phi_1 &= \{p\}\\ \Phi_2 &= \{q\}\\ \gamma_1 &= p \leftrightarrow q\\ \gamma_2 &= \neg(p \leftrightarrow q)\end{aligned}$$

What are the NE?

$\Rightarrow$ Some Boolean games have no NE.

Decision Problems

- **Membership**:
  Given a game G and strategy profile $\vec{\sigma}$, is $\vec{\sigma} \in NE(G)$?
- **Non-Emptiness**:
  Given a game G, is $NE(G) \neq \emptyset$?

NE Membership is co-NP-complete

Work with the complement problem, of verifying that some player has a beneficial deviation.

- Membership of NP: Guess a player i and strategy σ_i' and verify that i does better with σ_i' than their component of $\vec{\sigma}$.
- NP Hardness: Reduce SAT. Given SAT instance φ define 1-player game with $\gamma_1 = \varphi \wedge z$ where z is a new variable. Define strategy σ_1 which sets all variables to false. φ is then satisfiable iff i has a beneficial deviation from σ_1.

Non-Emptiness is Σ_2^p-complete

Membership

The game has an NE iff the following statement is true:

$$\exists \vec{\sigma} \bigwedge_{i \in N} \left(\vec{\sigma} \not\models \gamma_i \rightarrow (\forall \sigma'_i : (\vec{\sigma}_{-i}, \sigma'_i) \not\models \gamma_i)\right)$$

The statement above is an instance of $\text{QBF}_{2,\exists}$, whose satisfiability can be checked in Σ_2^p.

Non-Emptiness is Σ_2^p-complete

Hardness

Reduce $QBF_{2,\exists}$ to the problem of non-emptiness in a 2-player Boolean games.
Suppose $\exists X \forall Y \psi(X, Y)$ is the $QBF_{2,\exists}$ instance.
Define a game with:

- $\Phi_1 = X \cup \{x\}$ and $\gamma_1 = \psi(X, Y) \vee (x \leftrightarrow y)$
- $\Phi_2 = Y \cup \{y\}$ and $\gamma_2 = \neg\psi(X, Y) \wedge \neg(x \leftrightarrow y)$

Only NE if $\exists X \forall Y \psi(X, Y)$ is true.

Introducing Costs

Boolean Games with Costs

- Introduce costs to Boolean games: assigning a value to a variable induces a cost on the agent making the assignment.
- Preferences:
  - **Primary** aim is to achieve goals
  - **Secondary** aim is to minimise costs.
- Cost = energy requirements, time associated with actions...

Boolean Games with Costs

Formally, a Boolean game with costs is given by a structure

$$G = (N, \Phi_1, \dots, \Phi_n, \gamma_1, \dots, \gamma_n, c)$$

where $(N, \Phi_1, \dots, \Phi_n, \gamma_1, \dots, \gamma_n)$ is a Boolean game and

$$c : \Phi \times \mathbb{B} \to \mathbb{R}_{\geq}$$

is a **cost function**: $c(p, b)$ is the cost of assigning $b \in \mathbb{B}$ to p.

Let $c_i(\sigma_i)$ be the total cost of player i's choice σ_i:

$$c_i(\sigma_i) = \sum_{p \in \Phi_i} c(p, \sigma_i(p))$$

Utility Again

Given game G the utility to i of outcome $(\sigma_1, \ldots, \sigma_n)$ is given by:

$$u_i(\sigma_1, \ldots, \sigma_n) = \begin{cases} 1 + \mu_i - c_i(\sigma_i) & \text{if } (\sigma_1, \ldots, \sigma_n) \models \gamma_i \\ -c_i(\sigma_i) & \text{otherwise.} \end{cases}$$

where μ_i is the cost of the **most expensive choice** to i:

$$\mu_i = \max\{c_i(\sigma_i) \mid \sigma_i \in \Sigma_i\}$$

Utility Again

Given game G the utility to i of outcome $(\sigma_1, \ldots, \sigma_n)$ is given by:

$$u_i(\sigma_1, \ldots, \sigma_n) = \begin{cases} 1 + \mu_i - c_i(\sigma_i) & \text{if } (\sigma_1, \ldots, \sigma_n) \models \gamma_i \\ -c_i(\sigma_i) & \text{otherwise.} \end{cases}$$

where μ_i is the cost of the **most expensive choice** to i:

$$\mu_i = \max\{c_i(\sigma_i) \mid \sigma_i \in \Sigma_i\}$$

Properties:

1. an agent prefers all outcomes that satisfy its goal over all those that do not satisfy it;
2. between two outcomes that satisfy its goal, an agent prefers the one that minimises total cost; and
3. between two valuations that do not satisfy its goal, an agent prefers to minimise total cost.

Let's See Who is Awake…

- Suppose $\vec{\sigma}$ is an NE such that $\vec{\sigma} \not\models \gamma_i$. What can we say about player i's choice in $\vec{\sigma}$?

Let's See Who is Awake...

- Suppose $\vec{\sigma}$ is an NE such that $\vec{\sigma} \not\models \gamma_i$. What can we say about player *i*'s choice in $\vec{\sigma}$? Player *i*'s choice σ_i satisfies

$$\sigma_i \in \arg\min_{\sigma' \in \Sigma_i} c_i(\sigma')$$

- What is the largest utility a player can get?

Let's See Who is Awake...

- Suppose $\vec{\sigma}$ is an NE such that $\vec{\sigma} \not\models \gamma_i$. What can we say about player *i*'s choice in $\vec{\sigma}$? Player *i*'s choice σ_i satisfies

$$\sigma_i \in \arg\min_{\sigma' \in \Sigma_i} c_i(\sigma')$$

- What is the largest utility a player can get? $1 + \mu_i$
- What is the smallest utility a player can get?

Let's See Who is Awake...

- Suppose $\vec{\sigma}$ is an NE such that $\vec{\sigma} \not\models \gamma_i$. What can we say about player i's choice in $\vec{\sigma}$? Player i's choice σ_i satisfies

$$\sigma_i \in \arg\min_{\sigma' \in \Sigma_i} c_i(\sigma')$$

- What is the largest utility a player can get? $1 + \mu_i$
- What is the smallest utility a player can get? $-\mu_i$
- What is the smallest utility a player can get if they get their goal achieved?

Let's See Who is Awake...

- Suppose $\vec{\sigma}$ is an NE such that $\vec{\sigma} \not\models \gamma_i$. What can we say about player *i*'s choice in $\vec{\sigma}$? Player *i*'s choice σ_i satisfies

$$\sigma_i \in \arg\min_{\sigma' \in \Sigma_i} c_i(\sigma')$$

- What is the largest utility a player can get? $1 + \mu_i$
- What is the smallest utility a player can get? $-\mu_i$
- What is the smallest utility a player can get if they get their goal achieved? 1
- What is the largest utility a player can get if they don't get their goal achieved?

Let's See Who is Awake...

- Suppose $\vec{\sigma}$ is an NE such that $\vec{\sigma} \not\models \gamma_i$. What can we say about player *i*'s choice in $\vec{\sigma}$? Player *i*'s choice σ_i satisfies

$$\sigma_i \in \arg\min_{\sigma' \in \Sigma_i} c_i(\sigma')$$

- What is the largest utility a player can get? $1 + \mu_i$
- What is the smallest utility a player can get? $-\mu_i$
- What is the smallest utility a player can get if they get their goal achieved? 1
- What is the largest utility a player can get if they don't get their goal achieved? 0

An Example

Suppose:

$$
\begin{aligned}
\Phi_1 &= \{p\} \\
\Phi_2 &= \{q, r\} \\
\gamma_1 &= q \\
\gamma_2 &= q \vee r \\
& \text{All costs are 0}
\end{aligned}
$$

What are the NE?

An Example

Suppose:

$$
\begin{array}{rcl}
\Phi_1 & = & \{p\} \\
\Phi_2 & = & \{q, r\} \\
\gamma_1 & = & q \\
\gamma_2 & = & q \vee r \\
\\
c_2(q, \top) & = & 5 \\
c_2(q, \bot) & = & c_2(r, \top) = c_2(r, \bot) = 0 \\
& & \text{Other costs are } 0
\end{array}
$$

What are the NE?

What if there are more than one rounds?

Game-Theoretic Verification of Multi-Agent Systems[3]

Part III: LTL and Games

Muhammad Najib
Heriot-Watt University

The 24th European Agent Systems Summer School
(EASSS 2024)

[3] Adapted from lecture slides by Mike Wooldridge (`mjw@cs.ox.ac.uk`) and Julian Gutierrez (`Julian.Gutierrez@monash.edu`).

Iterated Boolean Games (iBG)

- A model of multi-agent systems in which players repeatedly choose truth values for Boolean variables under their control.
- Players behave selfishly in order to achieve individual **goals**.
- Goals expressed as **Linear Temporal Logic** (LTL) formulae.

Iterated Boolean games

An iBG is a structure

$$G = (N, \Phi, \Phi_1, \ldots, \Phi_n, \gamma_1, \ldots, \gamma_n)$$

where

- $N = \{1, \ldots, n\}$ is a set of **agents** (the players of the game),
- $\Phi = \{p, q, \ldots\}$ is a finite set of **Boolean variables**,
- $\Phi_i \subseteq \Phi$ is the set of variables controlled by player i,
- γ_i is the **LTL goal** of player i.

Models for LTL

- Let V be the set of **valuations** of Boolean variables Φ.
- Let V_i be the valuations for the variables Φ_i controlled by player i.
- Models of LTL formulae φ are **runs** ρ: infinite sequences in V^ω.
- We write $\rho \models \varphi$ to mean ρ satisfies LTL formula φ.

Playing an iBG

- Players play an infinite number of rounds, where on each round each player chooses values for their variables.
- The sequence of valuations traced out in this way forms a run, which either satisfies or doesn't satisfy a player's goal.
- A **strategy** for i is thus abstractly a function

$$f : V^* \to V_i$$

  ...but this isn't a **practicable** representation.
- So we model strategies as **finite state machines (FSM) with output** (transducers).

Machine strategies

A machine strategy for i is a structure:

$$\sigma_i = (Q_i, q_i^0, \delta_i, \tau_i)$$

where:

- Q_i is a finite, non-empty set of **states**,
- q_i^0 is the **initial** state,
- $\delta_i : Q_i \times V \to Q_i$ is a **state transition function**,
- $\tau_i : Q_i \to V_i$ is a **choice function.**

Strategy profiles

- A **strategy profile** $\vec{\sigma}$ is an n-tuple of machine strategies, one for each player i:

$$\vec{\sigma} = (\sigma_1, \ldots, \sigma_n).$$

- As strategies are **deterministic**, each strategy profile $\vec{\sigma}$ induces a unique run: $\rho(\vec{\sigma})$.

Nash Equilibrium

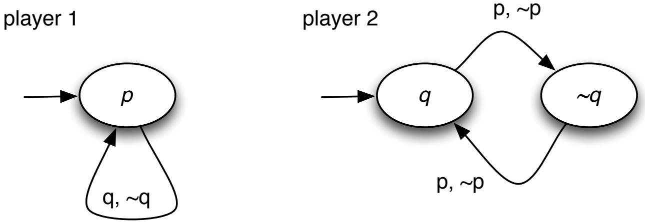
Strategy profile $\vec{\sigma} = (\sigma_1, \ldots, \sigma_i, \ldots, \sigma_n)$ is a (pure strategy) **Nash equilibrium** if for all players $i \in N$, if $\rho(\vec{\sigma}) \not\models \gamma_i$ then for all σ_i' we have

$$\rho(\sigma_1, \ldots, \sigma_i', \ldots, \sigma_n) \not\models \gamma_i$$

Let $NE(G)$ denote the Nash equilibria of a given iBG G.

An Example

- $N = \{1, 2\}$,
- $\Phi_1 = \{p\}$
- $\Phi_2 = \{q\}$
- $\gamma_1 = \mathbf{GF}(p \leftrightarrow q)$
- $\gamma_2 = \mathbf{GF}\neg(p \leftrightarrow q)$



These strategies form a NE.

Decision problems

MODEL CHECKING:
Given: Game G, strategy profile $\vec{\sigma}$, and LTL formula φ.
Question: Is it the case that $\rho(\vec{\sigma}) \models \varphi$?

MEMBERSHIP:
Given: Game G, strategy profile $\vec{\sigma}$.
Question: Is it the case that $\vec{\sigma} \in NE(G)$?

Decision problems

MODEL CHECKING:
Given: Game G, strategy profile $\vec{\sigma}$, and LTL formula φ.
Question: Is it the case that $\rho(\vec{\sigma}) \models \varphi$?

MEMBERSHIP:
Given: Game G, strategy profile $\vec{\sigma}$.
Question: Is it the case that $\vec{\sigma} \in NE(G)$?

Theorem

The MODEL CHECKING *and* MEMBERSHIP *problems are PSPACE-complete.*

Proof: follow from the fact that we can encode FSM strategies as LTL formulae.

Decision problems

E-NASH:
Given: Game G, LTL formula φ.
Question:$\exists \vec{\sigma} \in NE(G).\ \rho(\vec{\sigma}) \models \varphi$?

A-NASH:
Given: Game G, LTL formula φ.
Question: $\forall \vec{\sigma} \in NE(G).\ \rho(\vec{\sigma}) \models \varphi$?

NON-EMPTINESS:
Given: Game G.
Question: Is it the case that $NE(G) \neq \emptyset$?

Decision problems

E-NASH:
Given: Game G, LTL formula φ.
Question: $\exists \vec{\sigma} \in NE(G).\ \rho(\vec{\sigma}) \models \varphi$?

A-NASH:
Given: Game G, LTL formula φ.
Question: $\forall \vec{\sigma} \in NE(G).\ \rho(\vec{\sigma}) \models \varphi$?

NON-EMPTINESS:
Given: Game G.
Question: Is it the case that $NE(G) \neq \emptyset$?

Theorem

The E-NASH, A-NASH, *and* NON-EMPTINESS *problems are 2EXPTIME-complete.*

Proof: we can reduce **LTL synthesis** (Pnueli & Rosner, 1989)

Want to Find Out More?

- J. Gutierrez, M. Najib, G. Perelli, and M. Wooldridge. **On Computational Tractability for Rational Verification**. In **Proceedings of the Twenty Eighth International Joint Conference on Artificial Intelligence (IJCAI-2019)**. Macao, China, August 2019.
- J. Gutierrez, P. Harrenstein, M. Wooldridge. **From model checking to equilibrium checking: Reactive modules for rational verification**. In **Artificial Intelligence** 248:123–157, 2017.
- J. Gutierrez, P. Harrenstein, M. Wooldridge. **Reasoning about Equilibria in Game-like Concurrent Systems**. In **Annals of Pure and Applied Logic**, 168(2):373–403, February 2017.
- J. Gutierrez, P. Harrenstein, and M. Wooldridge. **Iterated Boolean Games**. In **Information & Computation**, 242:53–79, 2015.
- J. Gutierrez, P. Harrenstein, G. Perelli, and M. Wooldridge. **Nash Equilibrium and Bisimulation Invariance**. In **Proceedings of CONCUR-2017**, Berlin, Germany, September 2017.
- J. Paulin, A. Calinescu, and M. Wooldridge. **Understanding Flash Crash Contagion and Systemic Risk: A Micro-Macro Agent-Based Approach**. In **Journal of Economic Dynamics and Control**, Volume 100, Pages 200-229, March 2019.

Game-Theoretic Verification of Multi-Agent Systems[4]

Part IV: Reactive Modules Games

Muhammad Najib
Heriot-Watt University

The 24th European Agent Systems Summer School
(EASSS 2024)

[4] Adapted from lecture slides by Mike Wooldridge (`mjw@cs.ox.ac.uk`) and Julian Gutierrez (`Julian.Gutierrez@monash.edu`).

Reactive Modules Games

- iBGs are an abstraction of multi-agent systems, with some limitating assumptions (all players can choose any valuation for their variables)
- Practical model checkers use **high-level** model specification languages.
- **Reactive modules** is such a language:
  - a guarded command language for model specification
  - introduced by Alur & Henzinger in 1999
  - used in MOCHA, PRISM, ...

Reactive Modules

A multi-agent system is specified by a number of **modules** (=agents).

```
module toggle controls x
  init
  :: ⊤ ~> x' := ⊤;
  :: ⊤ ~> x' := ⊥;
  update
  :: x ~> x' := ⊥;
  :: ¬x ~> x' := ⊤;
```

A **module** has

1. an *interface*: name (*toggle*) and controlled variables (x)
2. a number of **init** and **update** *guarded commands* (::)

Reactive Module Arenas

An **arena** A is an $(n+2)$-tuple:

$$A = \langle N, \Phi, m_1, \ldots, m_n \rangle,$$

where:

- $N = \{1, \ldots, n\}$ is a set of agents
- Φ is a set of Boolean variables
- for each $i \in N$, $m_i = \langle \Phi_i, I_i, U_i \rangle$ is a module over Φ that defines the choices available to agent i.

Reactive Module Games

A reactive module game (RMG) is a tuple:

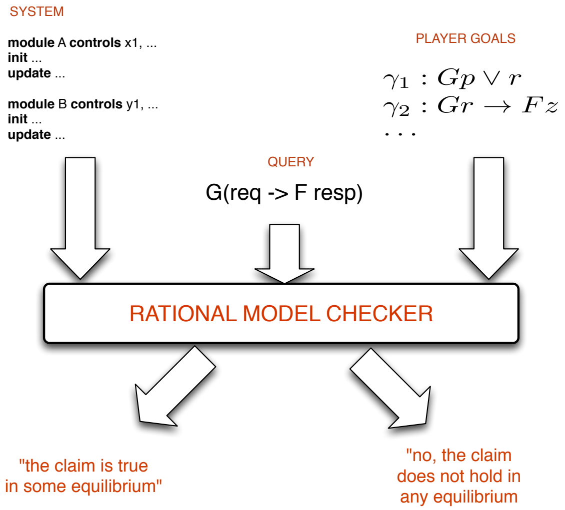
$$G = \langle A, \gamma_1, \ldots, \gamma_n \rangle$$

where:

- A is an arena
- for each player i in A, γ_i is the temporal logic goal of i.

Players choose **deterministic** FSM strategies
Deterministic strategies are **controllers**

Rational Verification in RMGs



LTL Reactive Module Games

NE-MEMBERSHIP

Given: *RMG G and strategy profile $\vec{\sigma}$.*

Question: *Is it the case that $\vec{\sigma} \in NE(G)$?*

Theorem

NE-MEMBERSHIP *for LTL RMGs is PSPACE-complete.*

NON-EMPTINESS

Given: *RMG G.*

Question: *Is it the case that $NE(G) \neq \emptyset$?*

Theorem

NON-EMPTINESS *for LTL RMGs is 2EXPTIME-complete, and it is 2EXPTIME-hard for 2-player games.*

LTL Reactive Module Games

E-NASH

Given: *RMG G, LTL formula* φ.

Question: *Does* $\rho(\vec{\sigma}) \models \varphi$ *hold for some* $\vec{\sigma} \in NE(G)$*?*

A-NASH

Given: *RMG G, LTL formula* φ.

Question: *Does* $\rho(\vec{\sigma}) \models \varphi$ *hold for all* $\vec{\sigma} \in NE(G)$*?*

Theorem

The E-NASH *and* A-NASH *problems for LTL RMGs are both 2EXPTIME-complete.*

Expressiveness

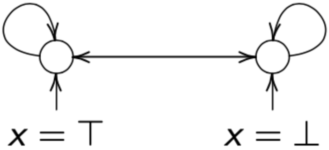
With respect iBGs, in general, RMGs may have different:

- Strategic power — different sets of available strategies
- Specification size — players' choices can be bounded

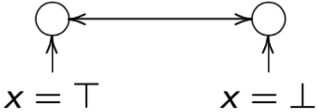
First difference: Strategic power

$$G_{\mathtt{iBG}} = (\{1\}, \{x\}, \gamma_1) \qquad vs \qquad G_{\mathtt{RML}} = (\{1\}, \{x\}, \gamma_1, toggle_1)$$

- Implicitly represented arena for the iBG:



- Succinctly represented arena for the RMG:



Second difference: Specification size

From $G = (\{1\}, \Phi = \{x, y\}, \Phi_1 = \{x, y\}, \gamma_1)$

To

$$
\begin{array}{l}
\texttt{module } G2RM \texttt{ controls } x, y \\
\quad \texttt{init} \\
\quad :: \ \top \texttt{~>} \ x' := \top \ ; \ y' := \top; \\
\quad :: \ \top \texttt{~>} \ x' := \top \ ; \ y' := \bot; \\
\quad :: \ \top \texttt{~>} \ x' := \bot \ ; \ y' := \top; \\
\quad :: \ \top \texttt{~>} \ x' := \bot \ ; \ y' := \bot; \\
\quad \texttt{update} \\
\quad :: \ \top \texttt{~>} \ x' := \top \ ; \ y' := \top; \\
\quad :: \ \top \texttt{~>} \ x' := \top \ ; \ y' := \bot; \\
\quad :: \ \top \texttt{~>} \ x' := \bot \ ; \ y' := \top; \\
\quad :: \ \top \texttt{~>} \ x' := \bot \ ; \ y' := \bot;
\end{array}
$$

We have $|G| = |\Phi| + |\gamma_1|$ and $|G2RM| = \mathcal{O}(2^{|\Phi|}) + |\gamma_1|$.

EVE: Verification Environment

`https://eve.cs.ox.ac.uk`

- We have implemented a tool for equilibrium checking RMGs.
- Takes as input:
  1. arena *A* specified in RML
  2. goals $\gamma_1, \ldots, \gamma_n$ for each player, specified in LTL
- computes NON-EMPTINESS, E-NASH and A-NASH problems
- combined parity games and automata-theoretic approach

Example: RMGs in EVE

Infinitely repeated matching pennies using RMGs:

module *alice* controls p
  init
  :: $\top$ ~> $p' := \top$;
  :: $\top$ ~> $p' := \bot$;
  update
  :: $\top$ ~> $p' := \top$;
  :: $\top$ ~> $p' := \bot$;
  goal
  :: $\mathbf{GF}(p \leftrightarrow q)$;

module *bob* controls q
  init
  :: $\top$ ~> $q' := \top$;
  :: $\top$ ~> $q' := \bot$;
  update
  :: $\top$ ~> $q' := \top$;
  :: $\top$ ~> $q' := \bot$;
  goal
  :: $\mathbf{GF}\neg(p \leftrightarrow q)$;

The SRML code of the above can be found here: `https://eve.cs.ox.ac.uk/examples/mp_example.txt`
Note that there are differences in the syntax used by EVE.
Try to run the code on EVE online

`https://eve.cs.ox.ac.uk/eve`

Exercise/Example 1

Design an RMG that has a Nash equilibrium, but such that the iBG over the same sets of controlled Boolean variables does not. Verify your solution using EVE.

Rule: You are not allowed to change the goals

Exercise/Example 1

Design an RMG that has a Nash equilibrium, but such that the iBG over the same sets of controlled Boolean variables does not. Verify your solution using EVE.

Rule: You are not allowed to change the goals

Idea:

Exercise/Example 1

Design an RMG that has a Nash equilibrium, but such that the iBG over the same sets of controlled Boolean variables does not. Verify your solution using EVE.

Rule: You are not allowed to change the goals

Idea:

- Design an iBG that has no NE

Exercise/Example 1

Design an RMG that has a Nash equilibrium, but such that the iBG over the same sets of controlled Boolean variables does not. Verify your solution using EVE.

Rule: You are not allowed to change the goals

Idea:

- Design an iBG that has no NE
- Specify in RMG

Exercise/Example 1

Design an RMG that has a Nash equilibrium, but such that the iBG over the same sets of controlled Boolean variables does not. Verify your solution using EVE.

Rule: You are not allowed to change the goals

Idea:

- Design an iBG that has no NE
- Specify in RMG
- “Restrict” the actions to introduce NE

Exercise/Example 1

Design an RMG that has a Nash equilibrium, but such that the iBG over the same sets of controlled Boolean variables does not. Verify your solution using EVE.

Rule: You are not allowed to change the goals

Idea:

- Design an iBG that has no NE
- Specify in RMG
- “Restrict” the actions to introduce NE

SRML code:

`https://eve.cs.ox.ac.uk/examples/mp_none.txt`

Exercise/Example 2

Consider a peer-to-peer network with 2 agents. At each time step, each agent either tries to download or to upload. In order for one agent to download successfully, the other must be uploading at the same time, and both are interested in downloading infinitely often.

Exercise/Example 2

Consider a peer-to-peer network with 2 agents. At each time step, each agent either tries to download or to upload. In order for one agent to download successfully, the other must be uploading at the same time, and both are interested in downloading infinitely often.

1. Use EVE to verify whether there **exists** a NE where both agents' goals are satisfied.

Exercise/Example 2

Consider a peer-to-peer network with 2 agents. At each time step, each agent either tries to download or to upload. In order for one agent to download successfully, the other must be uploading at the same time, and both are interested in downloading infinitely often.

1. Use EVE to verify whether there **exists** a NE where both agents' goals are satisfied.
2. Use EVE to verify whether in **all** NE, both agents' goals are satisfied.

Exercise/Example 2

Consider a peer-to-peer network with 2 agents. At each time step, each agent either tries to download or to upload. In order for one agent to download successfully, the other must be uploading at the same time, and both are interested in downloading infinitely often.

1. Use EVE to verify whether there **exists** a NE where both agents' goals are satisfied.
2. Use EVE to verify whether in **all** NE, both agents' goals are satisfied.
3. What modifications can be made so that query 2 above returns positively?

Exercise/Example 2

Consider a peer-to-peer network with 2 agents. At each time step, each agent either tries to download or to upload. In order for one agent to download successfully, the other must be uploading at the same time, and both are interested in downloading infinitely often.

1. Use EVE to verify whether there **exists** a NE where both agents' goals are satisfied.
2. Use EVE to verify whether in **all** NE, both agents' goals are satisfied.
3. What modifications can be made so that query 2 above returns positively? Without changing the goals...

Exercise/Example 2

Consider a peer-to-peer network with 2 agents. At each time step, each agent either tries to download or to upload. In order for one agent to download successfully, the other must be uploading at the same time, and both are interested in downloading infinitely often.

1. Use EVE to verify whether there **exists** a NE where both agents' goals are satisfied.
2. Use EVE to verify whether in **all** NE, both agents' goals are satisfied.
3. What modifications can be made so that query 2 above returns positively? Without changing the goals...

SRML code:

`https://eve.cs.ox.ac.uk/examples/p2p.txt`

A glimpse at the complexity: 2EXPTIME proof

Theorem

E-NASH *is 2EXPTIME-complete.*

Proof: Requires

- LTL synthesis (part 1),
- solving a collection of parity games (part 2),
- solving a product of Streett automata (part 3).

E-NASH complexity: Proof outline

- **Part 1:** A “standard” LTL to parity games reduction
  - From LTL formulae to Rabin automata on infinite trees
  - From deterministic Rabin automata on infinite trees to deterministic parity automata on infinite words
- **Part 2:** NE characterisation using parity games
  - From deterministic parity automata on infinite words to the construction of a multi-player parity game
  - Computing punishment regions in a collection of parity games
- **Part 3:** Definition of a path finding procedure over a product of deterministic Streett automata on infinite words

More about the complexity...

- Exponential in the size of the multi-agent system (SRML input).
- Exponential in the number of players, $|N|$.
- Doubly exponential in the size of the LTL goals in $\{\gamma_i\}_{i \in N}$.
- Doubly exponential in the size of the LTL specification/query φ.

E-NASH complexity: Proof outline – Part 1

- **Part 1:** A "standard" LTL to parity games reduction
  - From LTL formulae to Rabin automata on infinite trees
  - From deterministic Rabin automata on infinite trees to deterministic parity automata on infinite words

Theorem

Let $G = (M, \{\gamma_i\}_{i \in N})$ be an LTL game and $G' = (M', \{\alpha'_i\}_{i \in N})$ be its associated Parity game. Then, $NE(G) = NE(G')$.

Proof: Showing that for every strategy profile $\vec{\sigma}$ and player i, it is the case that $\rho(\vec{\sigma}) \models \gamma_i$ in M if and only if $\rho(\vec{\sigma}) \models \alpha'_i$ in M'.

ENASH complexity: Proof outline – Part 2

From Part 1 we get:

$$M' = A_M \times \prod_{i \in N} A_{\gamma_i}$$

- **Part 2:** NE characterisation using parity games
  - From deterministic parity automata on infinite words to the construction of a multi-player parity game
  - Computing punishment regions for several parity games.

$$\begin{tikzcd}
 & & & & & & & & s' \arrow[r, dash] & \sigma_i^{\mathsf{pun}\,j} \arrow[r] & \cdots \\
 & & & & & & & ((\vec{a}_k)_{-j}, a'_j) \arrow[ur] & & & \\
s_0 \arrow[r, dash] & \vec{a}_0 \arrow[r] & s_1 \arrow[r, dash] & \vec{a}_1 \arrow[r] & \cdots \arrow[r, dash] & \vec{a}_{k-1} \arrow[r] & s_k \arrow[r, dash] \arrow[ur, dash] & \vec{a}_k \arrow[r] & s_{k+1} \arrow[r, dash] & \vec{a}_{k+1} \arrow[r] & \cdots
\end{tikzcd}$$

Punishment region for player j: set of states in M' from which the coalition $i = N \setminus \{j\}$ can ensure that (has a strategy such that) player j does not get its parity goal α'_j satisfied.

E-Nash complexity: Proof outline – Part 2

- **Part 2:** NE characterisation using parity games
  - From deterministic parity automata on infinite words to the construction of a multi-player parity game
  - Computing punishment regions in a collection of parity games: For each $L \subseteq N$, compute M''_L from M', a game G''_L.

A path of M''_L that can be sustained in equilibrium by $\vec{\sigma}$ satisfies:

- all goals of players not in L and no goal for players in L, and
- *states*$(\rho(\vec{\sigma})) \subseteq \bigcap_{j \in L} \mathrm{Pun}_j$, if $L \neq \emptyset$, and
- *states*$(\rho(\vec{\sigma}_{-j}, \sigma'_j)) \subseteq \mathrm{Pun}_j$, for every $j \in L$ and σ'_j of j

ENASH complexity: Proof outline – Part 2

- **Part 2:** NE characterisation using parity games
  - From deterministic parity automata on infinite words to the construction of a multi-player parity game
  - Building punishment regions in a collection of parity games: For each $L \subseteq N$, compute M''_L from M' of G', a game G''_L.

Theorem

For all states s in M', we have $s \in Pun_j(G')$ iff $i = N \setminus \{j\}$ has a joint winning strategy against j in M''_L, for all $j \in L$ in G''_L.

Proof: Solution of $|2^N| - 1$ parity games (in quasipolynomial[5] time).

[5] Claude/Jain/Khoussainov/Li/Stephan, STOC'17.

E-NASH complexity: Proof outline – Part 3

From Part 2 we get M''_L and $\{\alpha'_i\}_{i \in N}$; and φ from Part 1.

- **Part 3:** Definition of a path finding procedure over a product of deterministic Streett automata on infinite words. Compute:
  - a Streett automaton recognising the paths of M''_L,
  - a Streett automaton recognising all paths satisfying φ in M''_L,
  - a Streett automaton for every parity function in $\{\alpha'_i\}_{i \in N \setminus L}$.

Check:

$$\mathcal{L}(\mathcal{S}_{M''_L} \times \mathcal{S}_\varphi \times \prod_{i \in N \setminus L} \mathcal{S}_{\alpha'_i}) \neq \emptyset$$

Streett automata are closed under conjunctions of Streett conditions; moreover, φ can be added to M''_L as the goal of a dummy player.

E-NASH complexity: Proof outline – Part 3

Defn: Action-run η is punishing-secure for j iff $states(\eta) \subseteq \text{Pun}_j$.

Theorem

For a Parity game G', there is a Nash Equilibrium strategy profile $\vec{\sigma} \in NE(G')$ such that $\pi(\vec{\sigma}) \models \varphi$ iff there is an ultimately periodic action-run η in G''_L such that, for every player $j \in L$, the run η is punishing-secure for j from state s^0, where π is the unique sequence of states generated by η from s^0 using $\vec{\sigma}$.

Proof: Showing that $\mathcal{L}(\mathcal{S}_{M''_L} \times \mathcal{S}_\varphi \times \prod_{i \in N \setminus L} \mathcal{S}_{\alpha'_i}) \neq \emptyset$ iff η is accepted.

Solving $\mathcal{L}(\mathcal{S}_{M''_L} \times \mathcal{S}_\varphi \times \prod_{i \in N \setminus L} \mathcal{S}_{\alpha'_i}) \neq \emptyset$ can be done in polynomial time because all automata have the same set of states and may differ only on its Streett condition.[6]

[6] Perrin/Pin, Infinite Words, Pure and Applied Mathematics, 2004.