Equilibrium Design for Concurrent Games

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

Monash University[1]

University of Kaiserslautern[2]

Sapienza University of Rome[3]

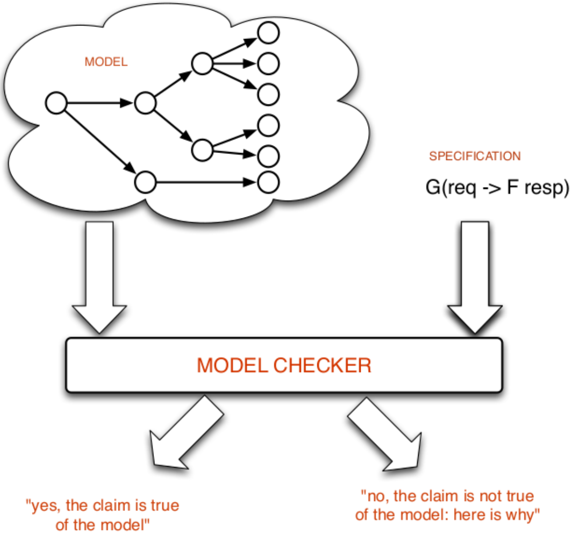
University of Oxford[4]

Model Checking in one slide

How to check system correctness.

- System represented as mathematical structure $\mathcal{K}$ (e.g., Kripke structure, Labeled transition system)
- Desired behavior represented as logic formula φ (e.g., Modal Logic, LTL, CTL, CTL*)
- The systems meets the behavior if (and only if) $\mathcal{K} \models \varphi$

Model Checking in one picture



Correctness Problem

- How do we define correctness in multi-agent systems?
- Each agent has their own goal. This implies:
  - Rationality
  - Strategic behaviour
  - Game theory as appropriate framework for correctness investigation

New standard of correctness

From Model Checking ...

Decide whether a given specification is satisfied over some/all executions of the (closed) system.

New standard of correctness

From Model Checking ...

Decide whether a given specification is satisfied over some/all executions of the (closed) system.

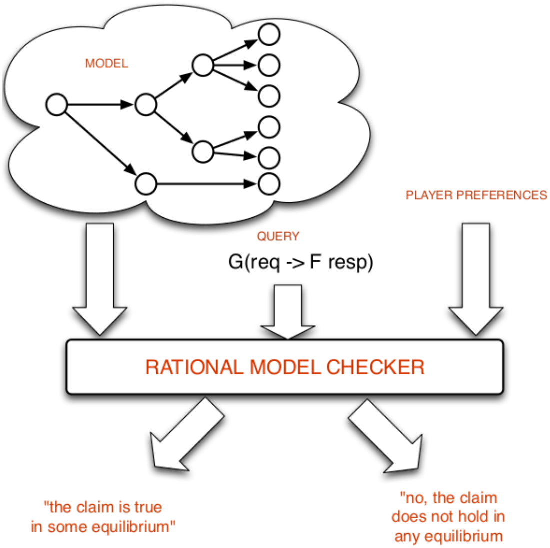
... to Equilibrium Checking!

Decide whether a given specification is satisfied over some/all **rational** executions of the (open) system.

Wooldridge et al. - Rational Verification: From Model Checking to Equilibrium Checking - AAAI'16

Kupferman et al. - Synthesis with Rational Environment - AMAI'16

Equilibrium Checking (in one slide)



Weighted Concurrent Game Models

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

$A = \langle \mathrm{N}, \mathrm{Ac}, \mathrm{St}, s_0, \mathrm{tr}, \lambda, (w_i)_{i \in \mathrm{N}} \rangle$

- 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;
- $w_i : \mathrm{St} \to \mathbb{Z}$ weight functions.

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

Weighted Concurrent Game Models

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

$A = \langle \mathrm{N}, \mathrm{Ac}, \mathrm{St}, s_0, \mathrm{tr}, \lambda, (w_i)_{i \in \mathrm{N}} \rangle$

- 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;
- $w_i : \mathrm{St} \to \mathbb{Z}$ weight functions.

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

Outcomes are infinite sequences of states and global actions

$\pi = s_0 \xrightarrow{\vec{a}_0} s_1 \xrightarrow{\vec{a}_1} \ldots \in (\mathrm{St} \times \mathrm{Ac}^{Ag})^{\omega}$

Agents' payoff

A payoff function pay_i for agent i is defined over outcomes

$$\mathsf{pay}_i : (\mathrm{St} \times \mathrm{Ac}^{Ag})^{\omega} \rightarrow \mathbb{R}$$

Agents' payoff

A payoff function pay_i for agent i is defined over outcomes

$$\mathsf{pay}_i : (\mathrm{St} \times \mathrm{Ac}^{Ag})^{\omega} \to \mathbb{R}$$

Temporal logic specification

$$\mathsf{pay}_i(\pi) = \begin{cases} 1, & \text{if } \pi \models \gamma_i \\ 0, & \text{if } \pi \not\models \gamma_i \end{cases},$$

$\gamma_i \in \mathsf{LTL}, \mathsf{GR}(1), \ldots$

Agents' payoff

A payoff function pay_i for agent i is defined over outcomes

$$\mathsf{pay}_i : (\mathrm{St} \times \mathrm{Ac}^{Ag})^{\omega} \to \mathbb{R}$$

Temporal logic specification

$$\mathsf{pay}_i(\pi) = \begin{cases} 1, & \text{if } \pi \models \gamma_i \\ 0, & \text{if } \pi \not\models \gamma_i \end{cases},$$

$\gamma_i \in \mathsf{LTL}, \mathsf{GR}(1), \ldots$

GR(1) specifications (fragment of LTL)

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

Agents' payoff

A payoff function pay_i for agent i is defined over outcomes

$$\text{pay}_i : (\mathrm{St} \times \mathrm{Ac}^{Ag})^\omega \to \mathbb{R}$$

Temporal logic specification

$$\text{pay}_i(\pi) = \begin{cases} 1, & \text{if } \pi \models \gamma_i \\ 0, & \text{if } \pi \not\models \gamma_i \end{cases},$$

$\gamma_i \in \mathrm{LTL}, \mathrm{GR}(1), \ldots$

GR(1) specifications (fragment of LTL)

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

Mean-Payoff

$$\text{pay}_i(\pi) = \liminf_{n \to \infty} \frac{1}{n} \sum_{j=0}^{n-1} w_i(\pi_j)$$

Agents' payoff

A payoff function pay_i for agent i is defined over outcomes

$$\text{pay}_i : (\mathrm{St} \times \mathrm{Ac}^{Ag})^{\omega} \to \mathbb{R}$$

Temporal logic specification

$$\text{pay}_i(\pi) = \begin{cases} 1, & \text{if } \pi \models \gamma_i \\ 0, & \text{if } \pi \not\models \gamma_i \end{cases},$$

$\gamma_i \in \text{LTL}, \text{GR}(1), \ldots$

GR(1) specifications (fragment of LTL)

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

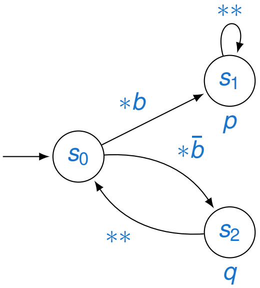
Mean-Payoff

$$\text{pay}_i(\pi) = \liminf_{n \to \infty} \frac{1}{n} \sum_{j=0}^{n-1} w_i(\pi_j)$$

Agents strategically try to maximise their payoff.

Some Examples: Qualitative Objectives

A qualitative game with $\mathrm{N} = \{\bigcirc, \blacksquare\}$. Actions for every $s_{x \in \{0,1,2\}}$, $\mathrm{Ac}_{\bigcirc}(s_x) = \{a, \bar{a}\}$ and $\mathrm{Ac}_{\blacksquare}(s_x) = \{b, \bar{b}\}$. Goals $\gamma_{\bigcirc} = \mathbf{FG}p$, $\gamma_{\blacksquare} = \mathbf{GF}q$.

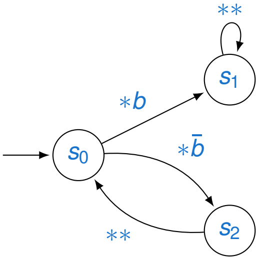


$\blacksquare$ wins by choosing the action $\bar{b}$ every time in s_0. Since $(s_0 s_2)^\omega \models \gamma_{\blacksquare}$, thus $\mathrm{pay}_{\blacksquare}((s_0 s_2)^\omega) = 1$ [a].

[a] Yes, I've made a slight abuse of notation here :|

Some Examples: Quantitative Objectives

A quantitative game with the same set of players and set of actions. Let $w_A(s_1) = 1, w_B(s_2) = 1$, and all zeros for the others.



Again, ■ "wins" by choosing the action $\bar{b}$ every time in s_0. She gets $\mathrm{pay}_{■}((s_0 s_2)^\omega) = \frac{1}{2}$, whilst ◯ gets $\mathrm{pay}_{◯}((s_0 s_2)^\omega) = 0$.

Strategies and Plays

Strategy

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

- Q, internal state (q_0 initial state);
- $\delta : Q \times \mathrm{St} \rightarrow Q$ internal transition function;
- $\tau : Q \rightarrow \mathrm{Ac}$ 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 $\mathcal{G} = \langle A, \text{pay}_1, \ldots, \text{pay}_{|N|} \rangle$ 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

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

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

Equilibrium Checking

Non-Emptiness

Given: Game $\mathcal{G}$.

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

Equilibrium Checking

Non-Emptiness

Given: Game $\mathcal{G}$.

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

E-Nash

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

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

Equilibrium Checking

Non-Emptiness

Given: Game $\mathcal{G}$.

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

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 $\mathcal{G}$?

Game types and complexities

Game type can be tuned using two different parameters:

(1) Temporal specification

(2) Players' goals.

Game types and complexities

Game type can be tuned using two different parameters:

(1) Temporal specification

(2) Players' goals.

Different types have different computational complexities.

Specification	Players' goals	Equilibrium checking
LTL	LTL	2EXPTIME-complete
LTL	GR(1)	PSPACE-complete
LTL	Mean-payoff	PSPACE-complete
GR(1)	GR(1)	FPT
GR(1)	Mean-payoff	NP-complete

Gutierrez et al. - Iterated Boolean Games - Inf&Comp'15

Gutierrez et al. - On Computational Tractability for Rational Verification - IJCAI'19

Dealing with missing 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.

Dealing with missing 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?

Dealing with missing 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

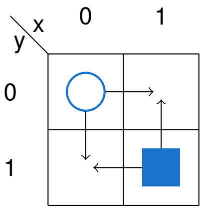
The problem with this is intrinsic in the system. Can we repair it in order to gain (desirable) equilibria?

Solution

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

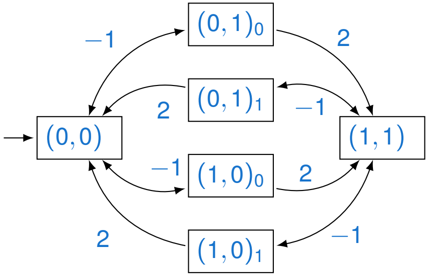
Almagor et al. - Repairing Multi-Player Games - CONCUR'15

Yet Another Example



Each time an agent moves one step, it gets payoff of -1. The goal of each agent is to visit each corners $(0, 0)$ and $(1, 1)$ in alternating fashion. To model this goal, we reward the robots with 2 units of energy, every time they travel from one corner to the opposite corner. Extra assumptions: at each timestep, each robot has to make a move, that is, it cannot stay at the same position for two consecutive timesteps, and each robot can only move at most one step.

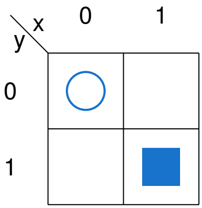
Yet Another Example: Converting into transition system



Transition system for player ◯. The vertices are marked with $(x, y)_f$, where $f \in \{0, 1\}$ is a flag to mark the last corner Player ◯ visited (0 for $(0, 0)$ and 1 for $(1, 1)$.) [a]

[a] Payoffs are on the edges instead of vertices, however, we can easily transform the transition system and push the payoffs to the vertices.

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



- Player ◯ moves: S, E, N, W,...; Player ■: N, W, S, E,... — this is a Nash equilibrium, each player gets $\frac{1}{2}$, and a **good** one.
- Player ◯ moves: S, E, W, N,...; Player ■: N, W, E, S,... — this is a Nash equilibrium, each player gets $\frac{1}{2}$, and a also **good** one.
- Player ◯ moves: S, E, N, W,...; Player ■: W, N, E, S,... — this is also a Nash equilibrium, with payoff of $\frac{1}{2}$ for each player, but a **bad** one.

How can we "nudge" the players such that the bad equilibria are eliminated—or good equilibrium introduced, if none exists?

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 $\kappa : \mathrm{N} \times \mathrm{St} \to \mathbb{N}$.

The cost of κ is $\mathrm{cost}(\kappa) = \sum_{i \in \mathrm{N}} \sum_{s \in \mathrm{St}} \kappa(i)(s)$.

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 $\kappa : \mathrm{N} \times \mathrm{St} \to \mathbb{N}$.
The cost of κ is $\mathsf{cost}(\kappa) = \sum_{i \in \mathrm{N}} \sum_{s \in \mathrm{St}} \kappa(i)(s)$.

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 $s \in \mathrm{St}$.

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 $\kappa : \mathrm{N} \times \mathrm{St} \to \mathbb{N}$.

The cost of κ is $\mathrm{cost}(\kappa) = \sum_{i \in \mathrm{N}} \sum_{s \in \mathrm{St}} \kappa(i)(s)$.

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 $s \in \mathrm{St}$.

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

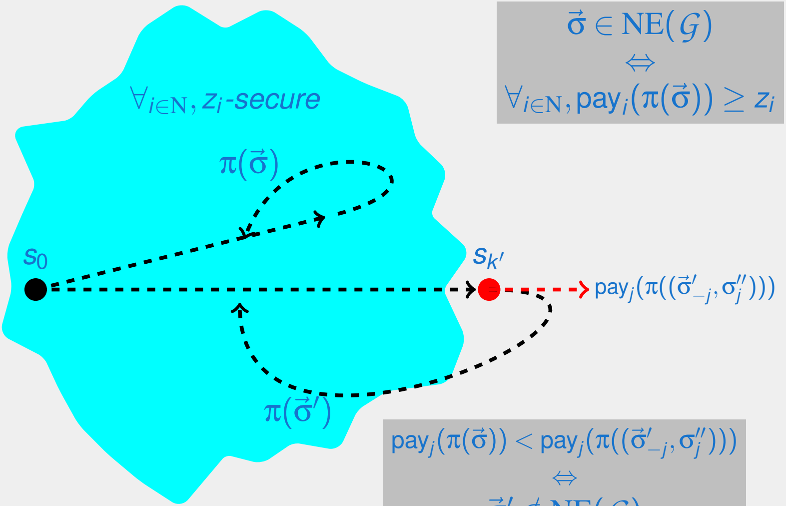
Definition (Strong Implementation)

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

Wooldridge et al. - Incentive engineering for Boolean games - AIJ'13

Theorem (NE characterisation)

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



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

Filling the toolbox

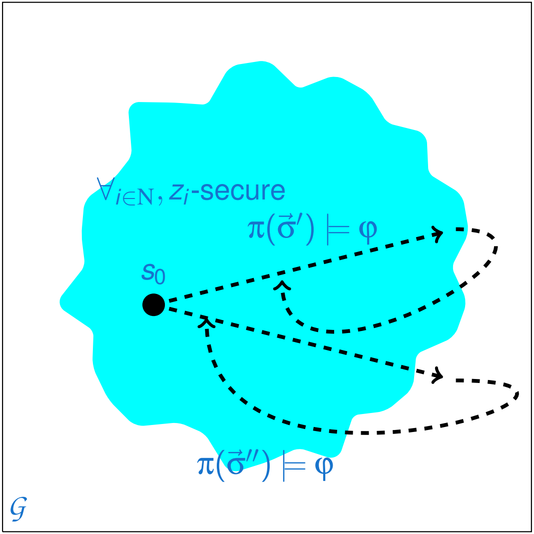
Theorem (Counting subsidy schemes)

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 $m = |\mathrm{St}| \cdot |\mathrm{N}|$

Solving Weak Implementation: Intuition

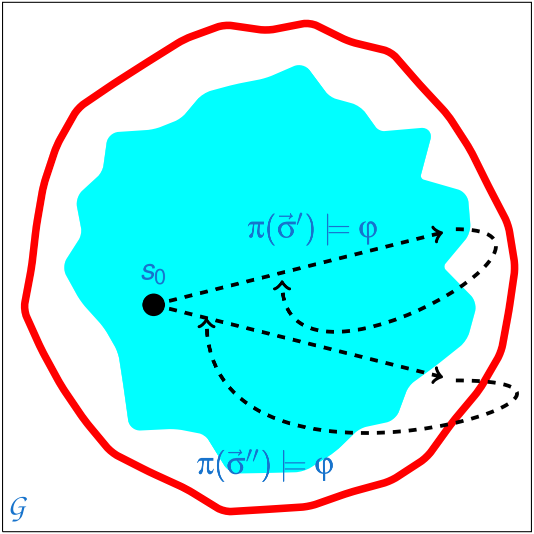


E-NASH:

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

No :-(

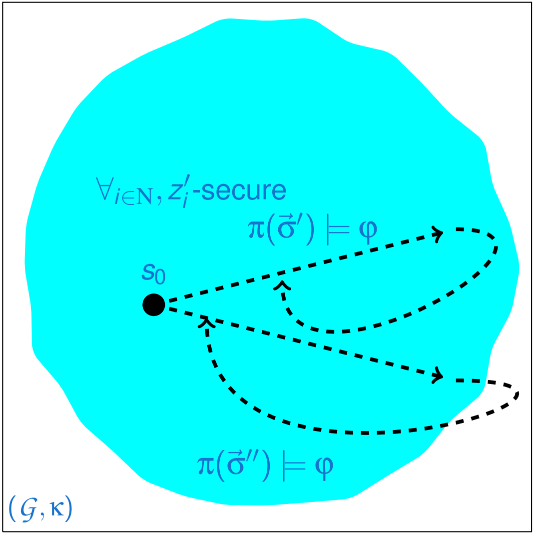
Solving Weak Implementation: Intuition



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

If only we can modify the perimeter...

Solving Weak Implementation: Intuition



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

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

Solving Weak Implementation: Algorithm

Algorithm Weak Implementation

Guess a subsidy scheme κ;

Guess a state $s \in \mathrm{St}$ for every player $i \in \mathrm{N}$, and

compute $z_i := \mathsf{val}_i(s)$ for every $i \in \mathrm{N}$ and $s \in \mathrm{St}$; *

Compute $(\mathcal{G}, \kappa)$;

Search for an ultimately periodic execution π in $(\mathcal{G}, \kappa)$ that satisfy φ and such that $z_i \leq \mathsf{pay}_i(\pi)$ for every $i \in \mathrm{N}$

Complexity

Solving Weak Implementation: Algorithm

Algorithm Weak Implementation

Guess a subsidy scheme κ;

Guess a state $s \in \mathrm{St}$ for every player $i \in \mathrm{N}$, and

compute $z_i := \mathsf{val}_i(s)$ for every $i \in \mathrm{N}$ and $s \in \mathrm{St}$; *

Compute $(\mathcal{G}, \kappa)$;

Search for an ultimately periodic execution π in $(\mathcal{G}, \kappa)$ that satisfy φ and such that $z_i \leq \mathsf{pay}_i(\pi)$ for every $i \in \mathrm{N}$

Complexity

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

Solving Weak Implementation: Algorithm

Algorithm Weak Implementation

Guess a subsidy scheme κ;

Guess a state $s \in \mathrm{St}$ for every player $i \in \mathrm{N}$, and

compute $z_i := \mathsf{val}_i(s)$ for every $i \in \mathrm{N}$ and $s \in \mathrm{St}$; *

Compute $(\mathcal{G}, \kappa)$;

Search for an ultimately periodic execution π in $(\mathcal{G}, \kappa)$ that satisfy φ and such that $z_i \leq \mathsf{pay}_i(\pi)$ for every $i \in \mathrm{N}$

Complexity

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

GR(1) Spec. Complexity Bounds

Upper Bound

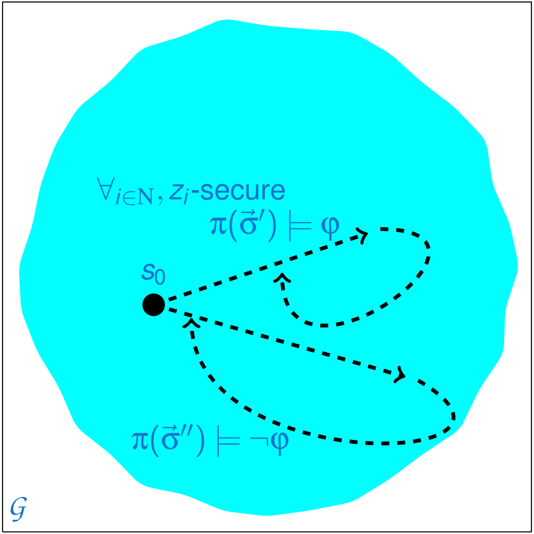
Recall that $\varphi = \bigwedge_{l=1}^{m} \mathbf{GF}\psi_l \rightarrow \bigwedge_{r=1}^{n} \mathbf{GF}\theta_r$

- $\mathrm{LP}(\psi_l)$ admits a solution if and only if there exists a path π in $\mathcal{G}$ such that $z_i \leq \mathrm{pay}_i(\pi)$ for every player i and visits $V(\psi_l)$ only *finitely many times*.
- $\mathrm{LP}(\theta_1, \ldots, \theta_n)$ admits a solution if and only if there exists a path π such that $z_i \leq \mathrm{pay}_i(\pi)$ for every player i and visits every $V(\theta_r)$ *infinitely many times*.
- there is a path π satisfying φ such that $z_i \leq \mathrm{pay}_i(\pi)$ for every player i in the game if and only if one of the two linear programs defined above has a solution.

Lower Bound

If $\varphi = \top$ and $\beta = 0$, then it's equivalent to checking the existence of Nash equilibrium in a mean-payoff game wich is NP-hard.

Solving Strong Implementation: Intuition

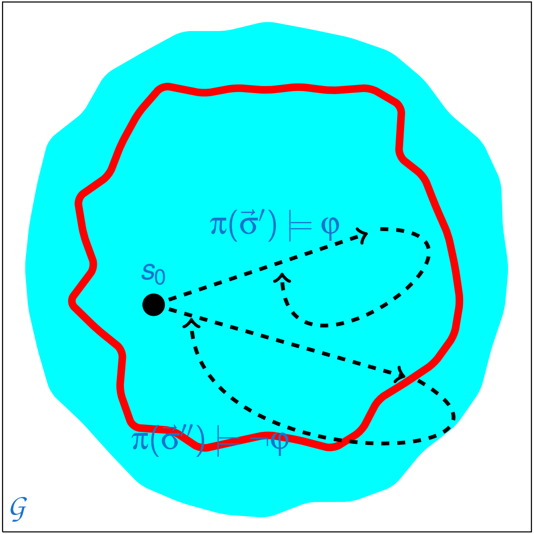


A-NASH:

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

No :-(

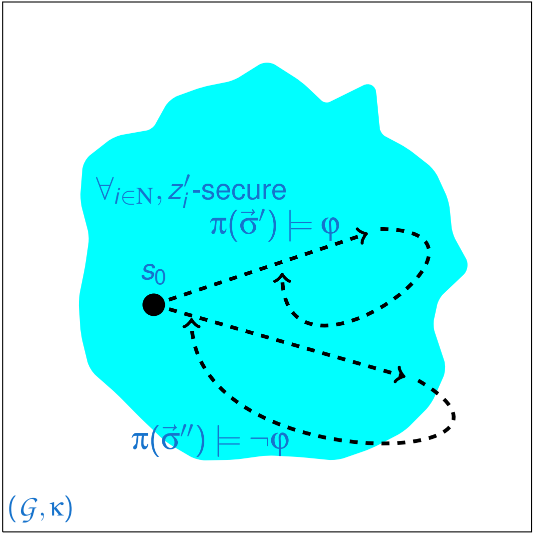
Solving Weak Implementation: Intuition



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

If only we can modify the perimeter...

Solving Weak Implementation: Intuition



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

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

Solving Strong Implementation

Observation

The Strong Implementation can be read as:

- There exists a subsidy scheme ... (existential guess)
- For all Nash Equilibria ... (universal guess)

Complexity

Solving Strong Implementation

Observation

The Strong Implementation can be read as:

- There exists a subsidy scheme ... (existential guess)
- For all Nash Equilibria ... (universal guess)

Complexity

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

Solving Strong Implementation

Observation

The Strong Implementation can be read as:

- There exists a subsidy scheme ... (existential guess)
- For all Nash Equilibria ... (universal guess)

Complexity

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

GR(1) Spec. Complexity Bounds

Upper Bound

Check whether the following expression is true

$$\exists \kappa, \tag{1}$$

$$\exists \vec{\sigma} \in \sigma_1 \times \cdots \times \sigma_n, \text{ such that } \vec{\sigma} \in \mathrm{NE}(\mathcal{G}, \kappa), \tag{2}$$

and

$$\forall \vec{\sigma}' \in \sigma_1 \times \cdots \times \sigma_n, \text{ if } \vec{\sigma}' \in \mathrm{NE}(\mathcal{G}, \kappa) \text{ then } \pi(\vec{\sigma}') \models \phi. \tag{3}$$

(2) can be checked in NP, (3) in coNP;

Lower Bound

Reduction to QSAT_2 (satisfiability of quantified Boolean formula with 2 alternations).

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.

Toolbox refill

Theorem

By setting $z_i = \max_{s \in \mathrm{St}} \mathsf{val}_i(s)$, *we have that:*

$$\beta_{\mathsf{OPT}} \leq \beta_{\mathsf{MAX}} = \sum_{i \in \mathrm{N}} z_i \cdot (|\mathrm{St}| - 1)$$

The optimal budget should be found within 0 and β_{MAX}

Solving Optimality

Observation

From previous slide, we employ binary search over the possible budgets and the weak/strong implementation routine as an oracle.

Weak Implementation Complexity

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

Strong Implementation Complexity

- For LTL specifications: PSPACE-complete (binary search is absorbed)
- For GR(1) specifications: $FP^{\Sigma_2^P}$-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

Conclusions and Future work

- Introduced the notion of Equilibrium Design for multi-agent games;
- Instantiated a new class of problems via Subsidy Schemes;
- Investigated on the complexity of these problems.
- Future work:
  - Optimising social welfare: fairer NE is desirable, e.g., ultimatum game. Relatively "reliable" NE (via Weak Implementation) without climbing polynomial hierarchy ladder.
  - Relatively low complexity class $\rightarrow$ practical implementation (e.g. extension of EVE: http://eve.cs.ox.ac.uk/)