Games for Verification and Synthesis: Admissibility in ω-Regular Games

Jeux pour la vérification et la synthèse: le cas de l'admissibilité pour des objectifs ω-réguliers

Mathieu Sassolas

Laboratoire d'Algorithmique, Complexité, et Logique
Université Paris-Est Créteil

Séminaire MeFoSyLoMa – February 7th, 2014

Outline

Outline

1. Introduction: Games for synthesis
   - From verification to control using games
   - The case of multi-player games
   - The (formal) setting
2. The complexity of admissible strategies
3. Wrap-up and perspectives

What is verification?

↪ Lots of different things! (Depending on who you ask)

- In its widest form: ensuring that a system behaves according to its specification.
- In the model-checking sense: ensuring that a system given by a formal model behaves according to a specification given by a formula: *Does* $\mathcal{A} \models \varphi$*?*
- Assumes the system is already in its final form and considers the environment.
- In reality the system is open and evolves in an environment: *Does* $(\mathcal{A}||env) \models \varphi$ *whatever env does?*

Verification of partially specified systems

- What about earlier stages of design?
- Can we help design the system?
- Can we do it automatically?

$\rightsquigarrow$ Synthesize $\mathcal{A}$ such that $\forall env$, $(\mathcal{A}||env) \models \varphi$.

- In practice, the system is already partially defined, and the environment is constrained:

$$\exists \mathcal{A} \in \mathit{Refinements}(\mathcal{A}_0), \forall env \in \mathit{Refinements}(env_0)$$
$$(\mathcal{A}||env) \models \varphi$$

$\hookrightarrow$ Let's rephrase:

$$\exists \sigma \text{ a strategy for } \mathcal{A}_0, \forall \tau \text{ a strategy for } env_0,$$
$$\mathit{traces}(\mathcal{A}_0^\sigma || env_0^\tau) \subseteq \varphi$$

And now it's a (two-player) game!

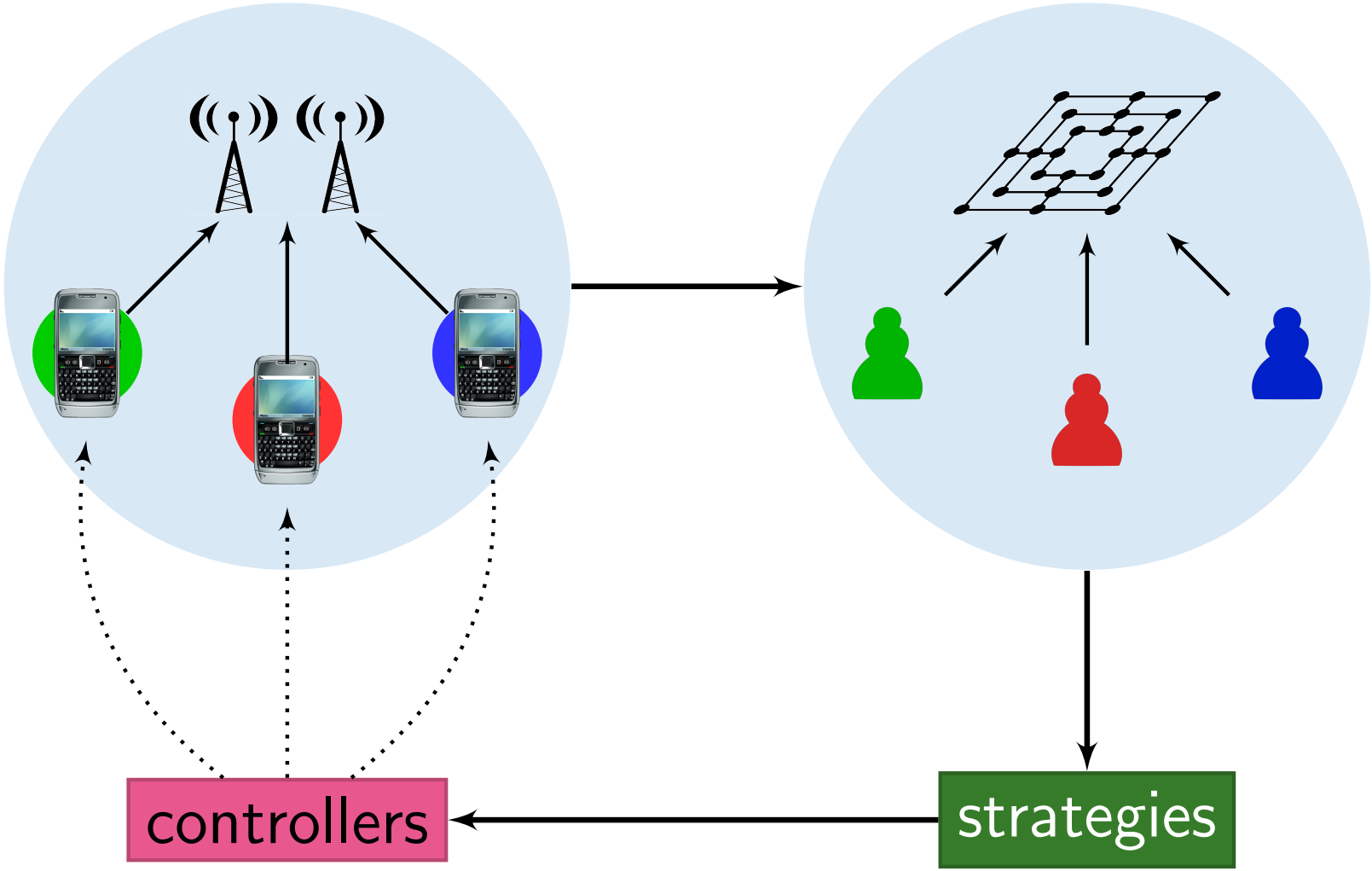
A theoretical note on two-player games

Theorem ([Martin, 75])

Two player perfect information games with Borelian objective are determined.

- Either the system or the environment has a winning strategy (provided φ is defined reasonably).
- That does not mean it is easy to determine who wins, nor that the winning strategy can be built.

The synthesis process



Introducing multi-player non zero-sum games

- The environment is actually other subsystems: $\mathcal{A}_1, \dots, \mathcal{A}_n$.
- All these subsystems also have a specification to comply to: $\varphi_1, \dots, \varphi_n$.
- It may be the case that the objectives of different subsystems can all be fulfilled at the same time (*vs* only one subsystem "wins" by fulfilling its specification).

Remark

These games are no longer determined: it is possible that no player is ensured to win.

What now?

Other paradigms are necessary to determine what strategies the players should play.

A little insight from economy theorists

- Economists have been playing with (possibly repeated) matrix games, *e.g.* battle of the sexes:

	theater	football
theater	$(6, 4)$	$(2, 3)$
football	$(0, 0)$	$(3, 7)$

- They want to model what is it for a player to be rational.

Remark

Humans are **not** rational.

Lucky for us...

Computers are rational.

Solution concepts

- A model of what it means to be rational is fixed.
- Every player knows that model and behaves accordingly.
- Every player also knows that the other players are rational and play accordingly.
- Every player knows that the other player knows. . .
- . . .

What outcomes of the games are to be expected?
Can this mechanism enforce some properties?

Models of rationality

Nash equilibrium "From a Nash equilibrium, I cannot obtain a higher payoff by changing only my strategy."

Subgame-perfect equilibrium A Nash equilibrium in every subgame.

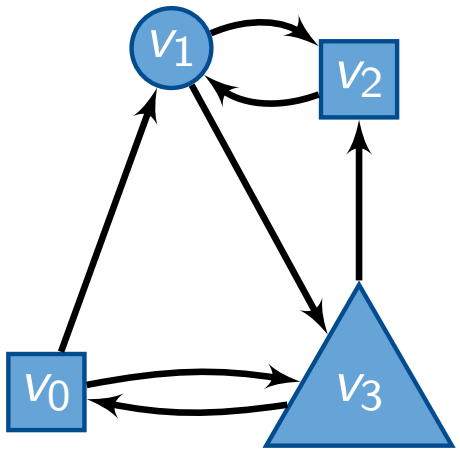
k-immune equilibrium "No coalition of size k players could increase its payoff by changing strategies."

Regret minimization "I play what minimizes the difference between the obtained payoff and what I could have obtained, had I known the strategies of other players."

Iterative elimination of dominated strategies "I dismiss strategies that are in all cases less efficient than another one; I then assume everyone does the same and start again."

Games on graph

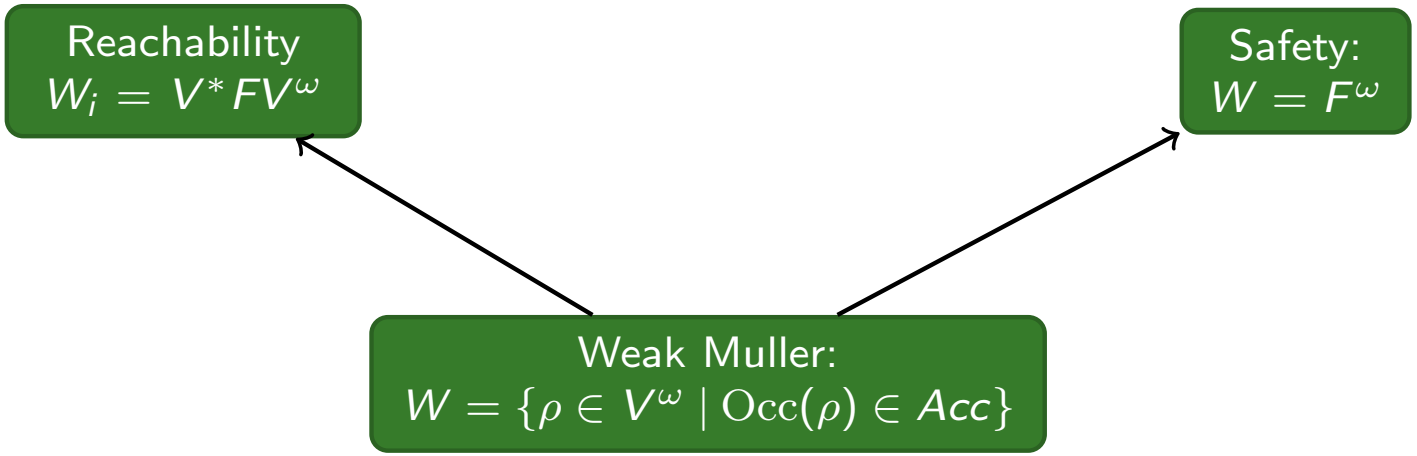
- Both the system and environment play on a common directed graph (with no end vertices).
- Each vertex belongs to one of the players.
- The player owning the vertex chooses the outgoing transition.
- The play goes on infinitely.
- Each player i has a set of winning runs: $W_i \subseteq V^\omega$.



Qualitative winning conditions
Finite horizon

Notations

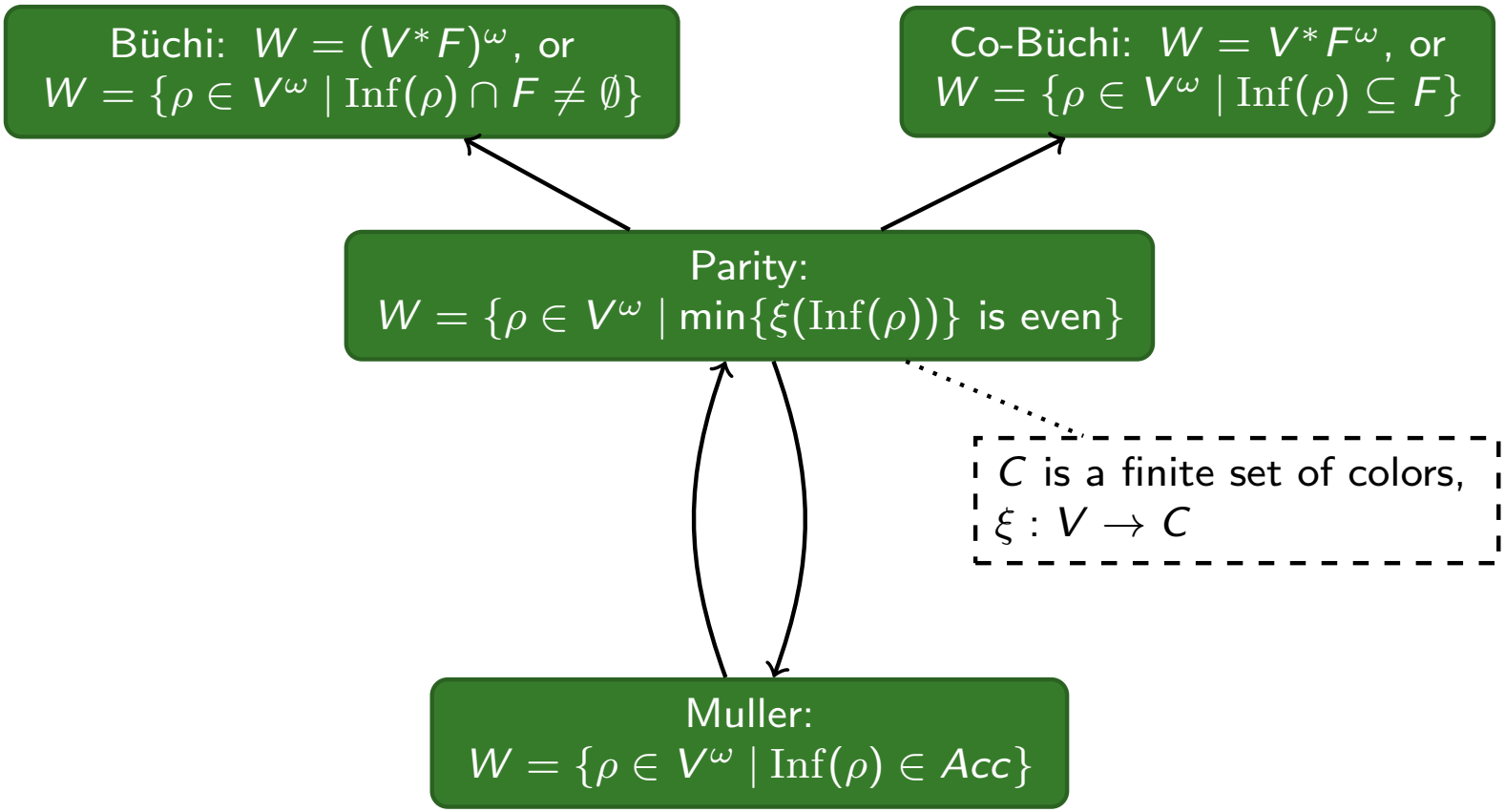
- $F \subseteq V$: set of final (*i.e.* "good") states.
- $\mathrm{Occ}(\rho)$: set of states occurring in ρ.
- $Acc \subseteq 2^V$ is the set of accepting sets of states.



Qualitative winning conditions
Prefix-independent

Notations

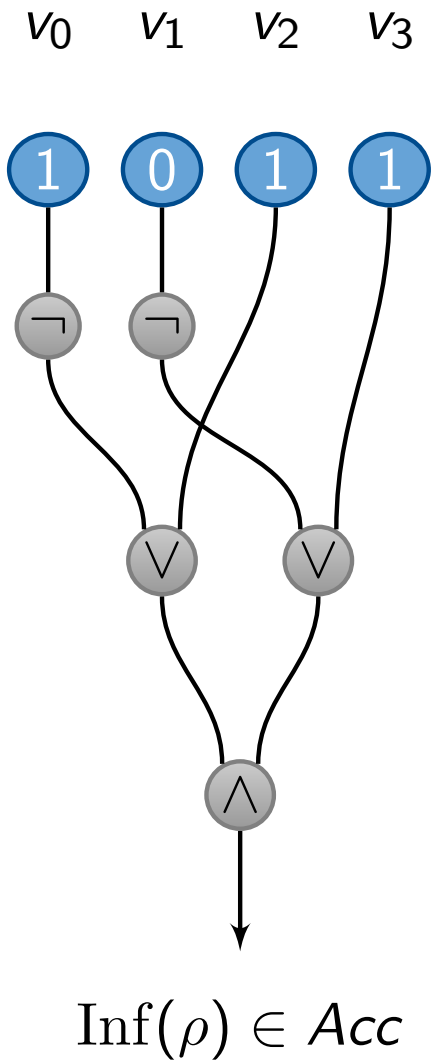
$\mathrm{Inf}(\rho)$: set of states visited infinitely often.



Circuit encoding of winning conditions

- Muller (resp. weak Muller) conditions require to know whether $\mathrm{Inf}(\rho) \in \mathit{Acc}$ (resp. $\mathrm{Occ}(\rho) \in \mathit{Acc}$).
- Encode *Acc* by a boolean circuit:
  - One input per vertex of V (set to 1 if $v \in \mathrm{Inf}(\rho)$ (resp. $\mathrm{Occ}(\rho)$).
  - Output is true if the encoded input set is a set of *Acc*.

$\hookrightarrow$ This representation is more concise, and makes it easier to combine conditions.



Quantitative winning conditions

- Edges are equipped with a payoff $w \in \mathbb{Z}$.
  - In the two-player zero-sum case, only one value, the opponent is assumed to have opposite value.
  - A player wins if the payoff of the run is above a given threshold.
- The value of a run for a player can be:
  - Total-payoff:

$$\lim_{n\to\infty} \sum_{0\leq k<n} w(\rho_k \to \rho_{k+1})$$

  - Mean-payoff:

$$\lim_{n\to\infty} \frac{\sum_{0\leq k<n} w(\rho_k \to \rho_{k+1})}{n}$$

  - Discounted-payoff: $(0 < \lambda < 1)$

$$\lim_{n\to\infty} \sum_{0\leq k<n} w(\rho_k \to \rho_{k+1}) \cdot \lambda^k$$

Complexity for the two-player case

- Reachability, Safety: polynomial time.
- Büchi, co-Büchi: polynomial time.
- Parity: NP ∩ coNP, suspected to be in P.
- Explicit Muller: polynomial time [Horn, 2008].
- Circuit Muller: PSPACE-complete [Hunter, 2007].
- Mean-payoff: NP ∩ coNP, suspected to be in P.

Outline

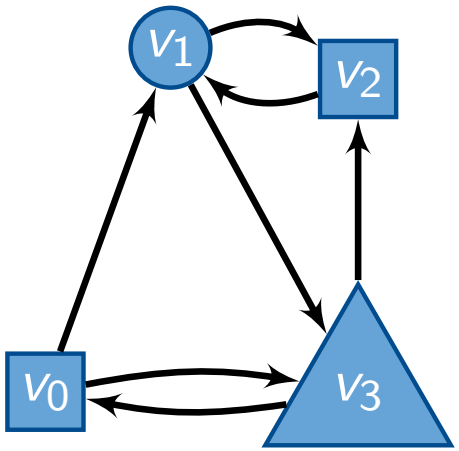
1. Introduction: Games for synthesis

2. The complexity of admissible strategies
   - Where are we again? a.k.a "The setting for the remainder of the talk"
   - Introducing values
   - Computing the values: the case of safety objectives
   - Computing the values: the case of prefix-independent objectives
   - A small case study: the metro system

3. Wrap-up and perspectives

Playing games

Multi-player non zero-sum infinite games on finite graphs



- The graph is divided between the players: $V = \biguplus_{i=1}^{n} V_i$.
- From a vertex in V_i, player i chooses the next state.
- One ω-regular winning condition, defined by a circuit, per player: $\varphi_1, \ldots, \varphi_n$.

- Each players plays according to a strategy $\sigma_i : V^* V_i \to V$ (that is assumed to respect the edges).
- The set of strategies is $\mathcal{S}$, $\mathcal{S}_i$ for the set of strategies of player i, $\mathcal{S}_{-i}$ for the set of strategies of all players but i.
- A strategy for each player (also called a profile) define a single outcome (starting from a state s): $Out_s(\sigma_1, \ldots, \sigma_n)$.

Dominance

Elimination of dominated strategies: the idea

"I dismiss strategies that are in all cases less efficient than another one; I then assume everyone does the same and start again."

Definition (Dominance)

Let S be a set of strategies. $\sigma_i' \succcurlyeq_S \sigma_i$ if, and only if, $\forall s \in V$,

$$\forall \tau \in S_{-i}, Out_s(\sigma_i, \tau) \vDash \varphi_i \Rightarrow Out_s(\sigma_i', \tau) \vDash \varphi_i.$$

- Strict dominance $\sigma_i' \succ_S \sigma_i$ if, and only if, $\sigma_i' \succcurlyeq_S \sigma_i$ and $\sigma_i \not\succcurlyeq_S \sigma_i'$
- Strategies that are not strictly dominated are admissible.

Iterative elimination

Elimination of dominated strategies: the idea

"I dismiss strategies that are in all cases less efficient than another one; I then assume everyone does the same and start again."

- Start with $\mathcal{S}_i^0 = \mathcal{S}_i$ and set for every player i
$$\mathcal{S}_i^{n+1} := \mathcal{S}_i^n \setminus \left\{ \sigma_i \;\middle|\; \exists \sigma_i' \in \mathcal{S}_i^n, \sigma_i' \succ_{\mathcal{S}^n} \sigma_i \right\}.$$
- Set of iteratively admissible strategies: $\mathcal{S}^* = \bigcap_{n \in \mathbb{N}} \mathcal{S}^n$

$\hookrightarrow$ Goal: compute $\mathcal{S}^*$ or at least decide properties thereof.

Remark

$\mathcal{S}^*$ is well defined and is reached after a finite number of iterations.

"Admissibility in Infinite Games" [Berwanger, STACS'07]

Decision problems on $\mathcal{S}^*$

The *winning coalition problem*

Given $W, L \subseteq P$, does there exists $\sigma_P \in \mathcal{S}^*$ such that all players of W win the game, and all players of L lose.

The *model-checking under admissibility problem*

Given φ an LTL formula, is it the case that for any profile $\sigma_P \in \mathcal{S}^*$, $Out(\sigma_P) \vDash \varphi$?

Values

Introduced in [Berwanger, STACS'07]

- If there is a winning strategy: value 1

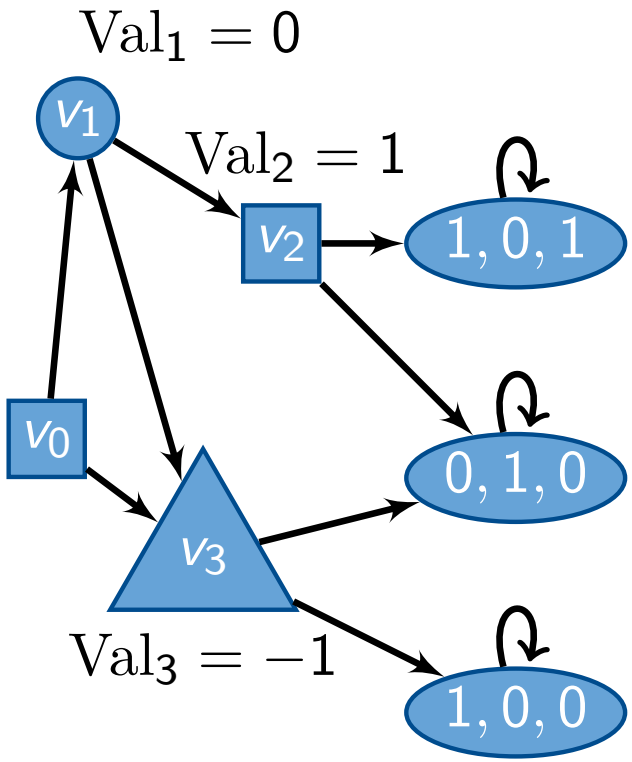
⇝ admissible strategies are the winning ones.

- It is impossible to win: value -1

⇝ all strategies are admissible.

- Otherwise: it is possible to win, but only with the help of others: value 0

↪ What are the admissible strategies in this case?



Values (ctd.)

Remarks

- A player should never decrease its own value.
- The value depends on $\mathcal{S}^n$ (*i.e.* the strategies available to the player itself and other players).

$\hookrightarrow$ How to compute those values?

- Note that when looking for a winning strategy ("is the value 1?"), it is now a two-player game (player i *vs* the rest of the world).
- And when looking for the absence of a winning outcome ("is the value -1?"), it is now a single-player game (everyone plays together).

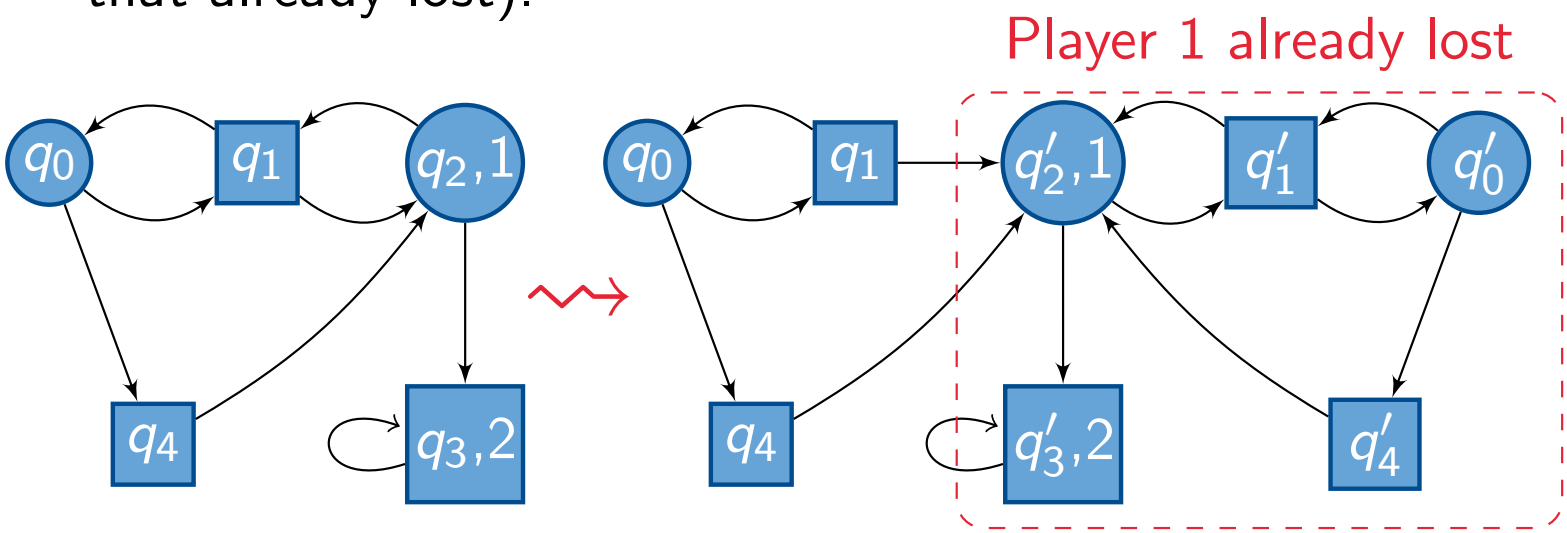
Safety games

- For each player, a set of states to avoid is given: Bad_i.
- Once Bad_i has been reached, player i has lost.
- Once a player has lost, every strategy is equally bad, hence no strategy dominates another.

Unfolding the graph

- For safety conditions, the existence of a winning strategy/winning collaboration depends only on:
  - the current state
  - whether the *Bad* state has already been visited
- The graph is *unfolded* to take this into account.

$\rightsquigarrow$ Yields a graph of size $|V| \times 2^{|P|}$, but with a structure (created by the inclusion partial order on the set of players that already lost).

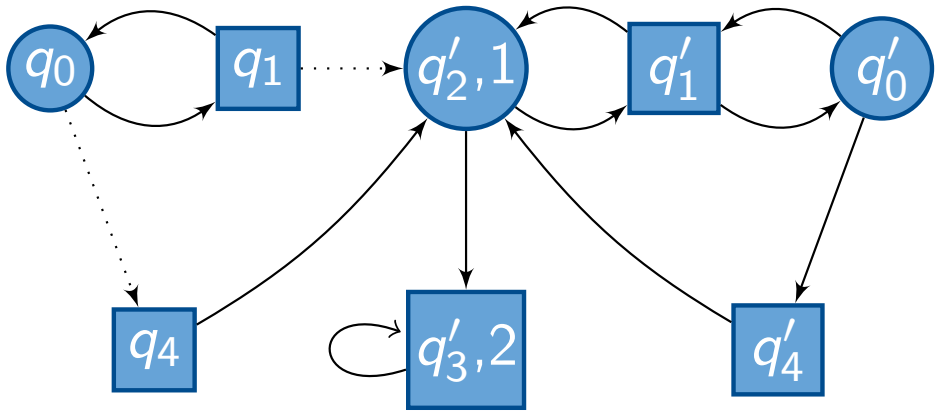


A local notion of dominance

- In unfolded safety games the rule to never decrease one's own value is sufficient for admissibility.
- At each iteration:
  - Compute the values for everyone, i.e.
    - ★ solve two-player safety games: i vs $P \setminus \{i\}$ (value 1)
    - ★ or one-player game (value -1)
  - Each player removes the edges that decrease its own value:

$$T_i^n = \{s \to s' \mid s \in V_i \text{ and } \mathrm{Val}_i^n(s) > \mathrm{Val}_i^n(s')\}\,.$$

  - Start again on this smaller game.



Complexity

- The previous algorithm on the unfolding yields (naively) an EXPTIME algorithm.
- But the structure of said unfolding allows recursive computation in PSPACE.
- Hardness: encoding of QSAT.

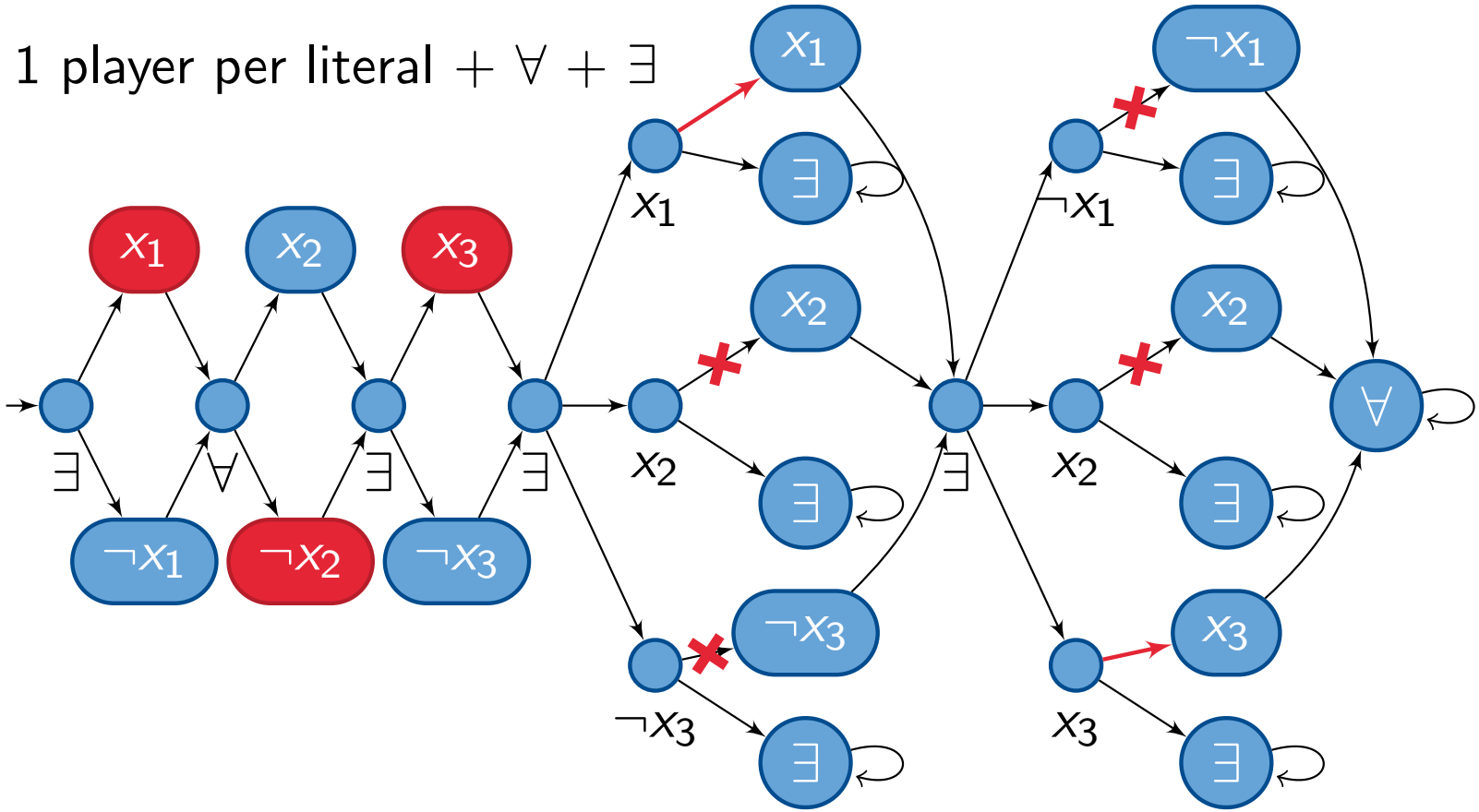
Theorem

The winning coalition problem with safety condition for each player is PSPACE*-complete.*

PSPACE-hardness of admissibility

Example of QSAT encoding

Is there an admissible profile such that $\exists$ wins?



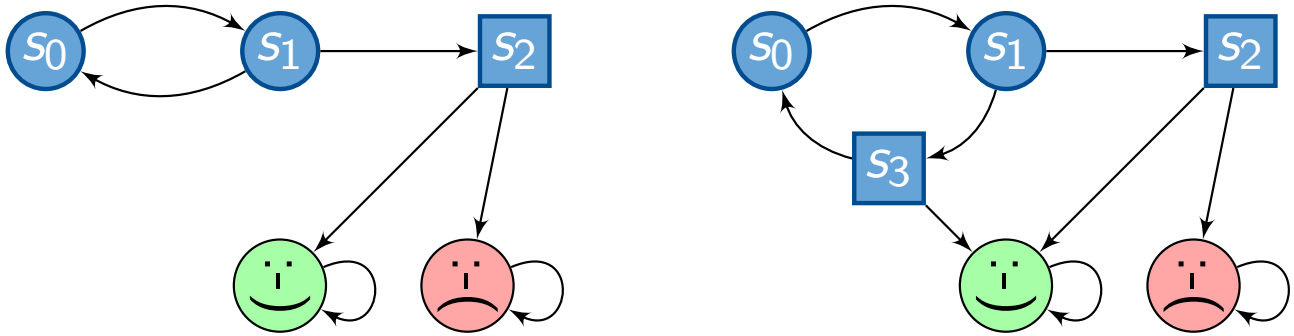
Game $\mathcal{G}_\mu$ with $\mu = \exists x_1 \forall x_2 \exists x_3 (x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3)$.

Prefix-independent objectives

- Winning or not does not depend on the prefix, but what happens in the long run.
- Comprises Büchi, co-Büchi, parity, Muller,...
- Is encoded by a circuit for each player.

Introducting fairness for admissibility

- Never decreasing one's own value is not sufficient to be admissible.
- In case the value is 0, need to allow other players to help.



- "Help!"-state for i: a state where j has several choices that are of value > -1 for i, while not changing the value for j.

$\rightsquigarrow$ Admissible strategies should be winning if the other players played fairly in those states.

Expressing $Out(\mathcal{S}^n)$ as an automaton

Admissible strategies are:

- the winning strategies if the value is 1;
- all strategies if the value is -1;
- strategies that either win or visit "Help!"-states infinitely often.

$\hookrightarrow$ These conditions can be translated into a circuit condition that accepts $Out(\mathcal{S}_i^{n+1}, \mathcal{S}^n_{-i})$: the outcomes of profiles where i plays a strategy of $\mathcal{S}_i^{n+1}$ and other players play with profile in $\mathcal{S}^n_{-i}$.

- Intersect all above conditions for every player: condition for $Out(\mathcal{S}^{n+1})$.
- Don't forget to forbid strict value decreasing!

Computing the values

- Value 1 is existence of a winning strategy assuming all players play in $\mathcal{S}^n$.
- This can be rephrased using $(Out(\mathcal{S}^n_j))_{j \in P}$, as a circuit condition.

$\rightsquigarrow$ Checking if the value is 1 is solving a two-player circuit game (PSPACE-complete).

- Value -1 is emptiness the winning runs that behave according to $\mathcal{S}^n$.
- This can be rephrased using $(Out(\mathcal{S}^n_j))_{j \in P}$, as a circuit condition.

$\rightsquigarrow$ Checking if the value is -1 is solving a single-player circuit game.

- Otherwise, the value is 0.

Complexity of the winning coalition problem

- Circuits allow concise composition.
- Solving the winning coalition problem also becomes a circuit emptiness check.

Theorem

The winning coalition problem with a circuit condition for each player is PSPACE*-complete.*

- For Büchi winning conditions, games needed to compute the values are parity games.

Theorem

The winning coalition problem with Büchi objectives is in P^{NP}*. Moreover, if there exists a polynomial algorithm for solving two-player parity games, the complexity reduces to* P*.*

Model-checking under admissibility

- Transform LTL formula ψ into a Büchi automaton $\mathcal{A}_{\neg\psi}$ that accepts runs that violate ψ.

$\rightsquigarrow$ It is of polynomial size.

- Intersect $\mathcal{A}_{\neg\psi}$ with the automaton for $Out(\mathcal{S}^*)$.
- Test emptiness.

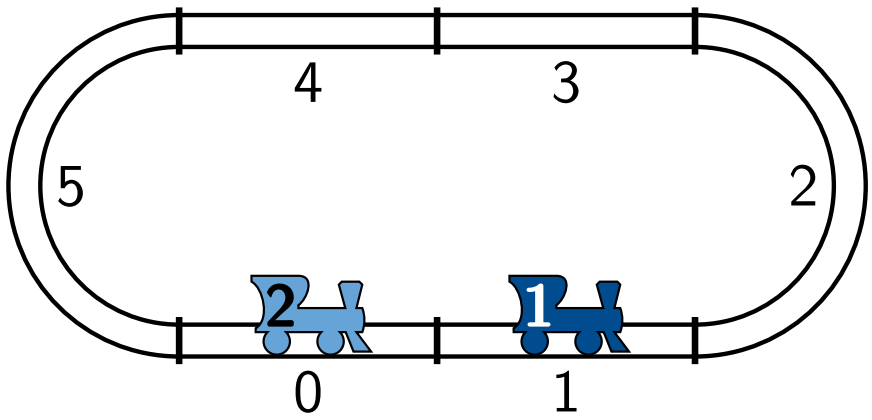
Theorem

The model-checking of LTL formula under admissibility with a circuit condition for each player and is PSPACE*-complete.*

Bonus

Since $Out(\mathcal{S}^*)$ has automata representation, other interesting problems we haven't yet thought about can be solved.

Context and arena



- At each step, all the trains successively declare whether they want to advance or not.
- Once everyone has chosen, all trains try to move synchronously.

- However, there is an evil environment that can prevent trains from moving.

↪ Some trains that wanted to move may in fact remain in their position.

⇝ All other trains must comply with their original choice.

Winning conditions

Trains To loop infinitely often.

Environment To create a collision.

Formula to model-check The absence of collision, $\psi_{\neg\text{coll}}$.

Remarks

- No player has a winning strategy alone.
- The formula $\psi_{\neg\text{coll}}$ is not verified on all paths of the model.
- The players are not *a priori* trying to satisfy $\psi_{\neg\text{coll}}$: indeed, the environment's objective is to negate $\psi_{\neg\text{coll}}$.

Results of iterative admissibility

- Iteratively admissible strategies for trains never try to advance if the track section ahead is not already empty.
- When this is possible, they will infinitely often choose to advance.
- The environment creates collisions whenever possible.
- It can also stop completely the progress of trains.

$\hookrightarrow$ This ensures no collision: $\mathit{Out}(\mathcal{S}^*) \models \psi_{\neg\mathrm{coll}}$.

$\rightsquigarrow$ This does not ensure that the trains will fulfill their objectives since they can be blocked indefinitely.

Outline

On admissibility

- Algorithms to compute the outcome of admissible strategies no harder (complexity-wise) than in the two-player case.
- As a byproduct: an automaton for $Out(\mathcal{S}^*)$.
- Promising concept:
  - More likely to be used by machines than humans.
  - Small case-study: iterative admissibility yields goals that where hidden in the specification.
- What next? $\rightsquigarrow$ quantitative setting.

On games

- Games provide an interesting framework for verification and synthesis.
- Different concepts of games can be adapted to different cases: multi-player *vs* single-player, quantitative *vs* qualitative,...
- Lots need to be done:
  - Is there a perfect solution concept for multi-player games played by computer systems?
  - Are solution concepts adapted/adaptable to the quantitative setting?
  - What happens when there is only partial observation?
  - What happens when we allow randomized strategies?
  - Can we use this to model security (and not only safety)?

Thank you

Any questions ?