Games for Verification and Synthesis: Admissibility in $\omega$-Regular Games

Jeux pour la vérification et la synthèse: le cas de l’admissibilité pour des objectifs $\omega$-réguliers

Mathieu Sassolas

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

Séminaire MeFoSyLoMa – February 7th, 2014
Outline

1. Introduction: Games for synthesis
2. The complexity of admissible strategies
3. Wrap-up and perspectives
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: $\mathcal{A} \vDash \varphi$?

- Assumes the system is already in its final form and considers the environment.

- In reality the system is open and evolves in an environment: $\mathcal{A} \parallel \text{env} \vDash \varphi$ whatever env does?
What about earlier stages of design?
Can we help design the system?
Can we do it automatically?

\[ \text{Synthesize } A \text{ such that } \forall env, (A||env) \models \varphi. \]

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

\[ \exists A \in \text{Refinements}(A_0), \forall env \in \text{Refinements}(env_0) (A||env) \models \varphi \]

Let’s rephrase:

\[ \exists \sigma \text{ a strategy for } A_0, \forall \tau \text{ a strategy for } env_0, \text{traces}(A_0^\sigma||env_0^\tau) \subseteq \varphi \]

And now it’s a (two-player) game!
Theorem ([Martin, 75])

Two player perfect information games with Borelian objective are determined.

- Either the system or the environment has a winning strategy (provided \( \varphi \) 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

controllers

strategies
The environment is actually other subsystems: $\mathcal{A}_1, \ldots, \mathcal{A}_n$.

All these subsystems also have a specification to comply to: $\varphi_1, \ldots, \varphi_n$.

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:

<table>
<thead>
<tr>
<th></th>
<th>theater</th>
<th>football</th>
</tr>
</thead>
<tbody>
<tr>
<td>theater</td>
<td>(6, 4)</td>
<td>(2, 3)</td>
</tr>
<tr>
<td>football</td>
<td>(0, 0)</td>
<td>(3, 7)</td>
</tr>
</tbody>
</table>

- 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.
- $\text{Occ}(\rho)$: set of states occurring in $\rho$.
- $\text{Acc} \subseteq 2^V$ is the set of accepting sets of states.

#### Reachability

$W_i = V^* F V^\omega$

#### Safety

$W = F^\omega$

#### Weak Muller

$W = \{\rho \in V^\omega | \text{Occ}(\rho) \in \text{Acc}\}$
Qualitative winning conditions
Prefix-independent

Notations

Inf(\(\rho\)): set of states visited infinitely often.

Büchi: \(W = (V^*F)^\omega\), or \(W = \{\rho \in V^\omega \mid \text{Inf}(\rho) \cap F \neq \emptyset\}\)

Co-Büchi: \(W = V^*F^\omega\), or \(W = \{\rho \in V^\omega \mid \text{Inf}(\rho) \subseteq F\}\)

Parity: \(W = \{\rho \in V^\omega \mid \min\{\xi(\text{Inf}(\rho))\}\ \text{is even}\}\)

Muller: \(W = \{\rho \in V^\omega \mid \text{Inf}(\rho) \in \text{Acc}\}\)

\(C\) is a finite set of colors, \(\xi : V \rightarrow C\)
Muller (resp. weak Muller) conditions require to know whether $\text{Inf}(\rho) \in \text{Acc}$ (resp. $\text{Occ}(\rho) \in \text{Acc}$).

Encode $\text{Acc}$ by a boolean circuit:
- One input per vertex of $V$ (set to 1 if $v \in \text{Inf}(\rho)$ (resp. $\text{Occ}(\rho)$).
- Output is true if the encoded input set is a set of $\text{Acc}$.

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:**
    \[
    \lim_{n \to \infty} \sum_{0 \leq k < n} w(\rho_k \rightarrow \rho_{k+1})
    \]
  - **Mean-payoff:**
    \[
    \lim_{n \to \infty} \frac{\sum_{0 \leq k < n} w(\rho_k \rightarrow \rho_{k+1})}{n}
    \]
  - **Discounted-payoff:** $(0 < \lambda < 1)$
    \[
    \lim_{n \to \infty} \sum_{0 \leq k < n} w(\rho_k \rightarrow \rho_{k+1}) \cdot \lambda^k
    \]
Games and Admissibility
Mathieu Sassolas
LACL
2014-02-07

Games for synthesis
Why games?
Going multi-player

The setting
Admissibility
Conclusion

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.
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
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 \( \omega \)-regular winning condition, defined by a circuit, per player: \( \varphi_1, \ldots, \varphi_n \).

Each player plays according to a **strategy** \( \sigma_i : V^* V_i \to V \) (that is assumed to respect the edges).

The set of strategies is \( S, S_i \) for the set of strategies of player \( i \), \( S_{-i} \) 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 \( s \)): \( \text{Out}_s(\sigma_1, \ldots, \sigma_n) \).
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. $\sigma'_i \succ_S \sigma_i$ if, and only if, $\forall s \in V$, $\forall \tau \in S_{-i}$, $Out_s(\sigma_i, \tau) \models \varphi_i \Rightarrow Out_s(\sigma'_i, \tau) \models \varphi_i$.

- Strict dominance $\sigma'_i \succ_S \sigma_i$ if, and only if, $\sigma'_i \succ_S \sigma_i$ and $\sigma_i \not\succ_S \sigma'_i$.
- 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 $S_i^0 = S_i$ and set for every player $i$
  $$S_i^{n+1} := S_i^n \setminus \{\sigma_i \mid \exists \sigma'_i \in S_i^n, \sigma'_i \succeq S^n \sigma_i\}.$$  
- Set of iteratively admissible strategies: $S^* = \bigcap_{n \in \mathbb{N}} S^n$

Goal: compute $S^*$ or at least decide properties thereof.

Remark

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

“Admissibility in Infinite Games” [Berwanger, STACS’07]
**Decision problems on $S^*$**

**The winning coalition problem**

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

**The model-checking under admissibility problem**

Given $\varphi$ an LTL formula, is it the case that for any profile $\sigma_P \in S^*$, $Out(\sigma_P) \models \varphi$?
Values
Introduced in [Berwanger, STACS’07]

If there is a winning strategy: value 1

\[ \rightarrow \text{admissible strategies are the winning ones.} \]

It is impossible to win: value \(-1\)

\[ \rightarrow \text{all strategies are admissible.} \]

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

\[ \leftarrow \text{What are the admissible strategies in this case?} \]

\[ V_0 = 1, 0, 0 \]
\[ V_1 = \text{Val}_1 = 0 \]
\[ V_2 = 1, 0, 1 \]
\[ V_3 = \text{Val}_2 = 1 \]
\[ \text{Val}_3 = -1 \]
Remarks

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

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: \( \text{Bad}_i \).
- Once \( \text{Bad}_i \) 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.

\[ \Rightarrow \] 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).

- Player 1 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:
    $$T^n_i = \{ s \rightarrow s' \mid s \in V_i \text{ and } \text{Val}^n_i(s) > \text{Val}^n_i(s') \}.$$  
  - Start again on this smaller game.
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?

1 player per literal $+ \forall + \exists$

Game $G_\mu$ with $\mu = \exists x_1 \forall x_2 \exists x_3 (x_1 \lor x_2 \lor \neg x_3) \land (\neg x_1 \lor x_2 \lor x_3)$. 

Game $G_\mu$ with $\mu = \exists x_1 \forall x_2 \exists x_3 (x_1 \lor x_2 \lor \neg x_3) \land (\neg x_1 \lor x_2 \lor x_3)$. 

Graphical representation of the game $G_\mu$. 

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

Admissible strategies should be winning if the other players played fairly in those states.
Expressing $Out(S^n)$ 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.

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

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

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

$\implies$ 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 $S^n$.
- This can be rephrased using $(Out(S^n_j))_{j \in P}$, as a circuit condition.

$\implies$ 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 $\psi$ into a Büchi automaton $A_{\neg \psi}$ that accepts runs that violate $\psi$.

$\Rightarrow$ It is of polynomial size.

- Intersect $A_{\neg \psi}$ with the automaton for $Out(S^*)$.

- Test emptiness.

**Theorem**

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

**Bonus**

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

Trains

Environment

Formula to model-check
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.

\[ Out(S^*) \models \psi_{\neg\text{coll}}. \]

This ensures no collision: \( Out(S^*) \models \psi_{\neg\text{coll}}. \)

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

1. Introduction: Games for synthesis
2. The complexity of admissible strategies
3. Wrap-up and perspectives
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 $Out(S^*)$.
- Promising concept:
  - More likely to be used by machines than humans.
  - Small case-study: iterative admissibility yields goals that were hidden in the specification.
- What next? $\leadsto$ quantitative setting.
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)?
Any questions?