Tradeoffs in Infinite Games

Martin Zimmermann

Saarland University

May 15th, 2018

Scientific Talk
in the Habilitation Process
Church’s Synthesis Problem

Church 1957: Given a specification on the input/output behavior of a circuit (in some suitable logical language), decide whether such a circuit exists, and, if yes, compute one.
**Church’s Synthesis Problem**

Example

Interpret input \( i_j = 1 \) as client \( j \) requesting a shared resource and output \( o_j = 1 \) as the corresponding grant to client \( j \).

Typical properties:

1. If there are infinitely many requests of client \( j \), then also infinitely many grants for client \( j \).
2. At most one grant at a time (mutual exclusion).
3. No spurious grants.
Church’s Synthesis Problem

Solved by Büchi & Landweber in 1969.

**Insight:** Problem can be expressed as two-player game of infinite duration between the environment (producing inputs) and the circuit (producing outputs).
Consider the one-client case!
Consider the one-client case!

Now, the winning plays for the circuit player have to satisfy

1. if $i$ is visited infinitely often, then $s$ as well, and

2. $\ell$ is never visited.

Equivalently: color the vertices by natural numbers as above and require that almost all odd colors are followed by a larger even one. This is the classical parity condition for $\omega$-automata.
Consider the one-client case!

Now, the winning plays for the circuit player have to satisfy

1. if $i$ is visited infinitely often, then $s$ as well, and

2. $\ell$ is never visited.

Equivalently: color the vertices by natural numbers as above and require that almost all odd colors are followed by a larger even one.

This is the classical parity condition for $\omega$-automata.
Consider the one-client case!

Input: 0
Output:
Consider the one-client case!

Input: 0
Output: 0

Now, the winning plays for the circuit player have to satisfy
1. if $i$ is visited infinitely often, then $s$ as well, and
2. $\ell$ is never visited.

Equivalently: color the vertices by natural numbers as above and require that almost all odd colors are followed by a larger even one.

This is the classical parity condition for $\omega$-automata.
Consider the one-client case!

Input: 0 0
Output: 0
Consider the one-client case!

Input: 0 0
Output: 0 0
Consider the one-client case!

Input: 0 0 0
Output: 0 0
Consider the one-client case!

Input: 0 0 0
Output: 0 0 1
Consider the one-client case!

Input: 0 0 0 1
Output: 0 0 1
Consider the one-client case!

Input: 0 0 0 1
Output: 0 0 1 1
Consider the one-client case!

Input: 0 0 0 1 1 1
Output: 0 0 1 1

Now, the winning plays for the circuit player have to satisfy
1. if \( s \) is visited infinitely often, then \( s \) as well, and
2. \( \ell \) is never visited.

Equivalently:
- color the vertices by natural numbers as above and
- require that almost all odd colors are followed by a larger even one.

This is the classical parity condition for \( \omega \)-automata.
Consider the one-client case!

Input: 0 0 0 1 1
Output: 0 0 1 1 1

Now, the winning plays for the circuit player have to satisfy
1. if \( i \) is visited infinitely often, then \( s \) as well, and
2. \( \ell \) is never visited.

Equivalently:
- color the vertices by natural numbers as above and
- require that almost all odd colors are followed by a larger even one.

This is the classical parity condition for \( \omega \)-automata.
Consider the one-client case!

Input: 0 0 0 1 1 ···
Output: 0 0 1 1 1 ···

Now, the winning plays for the circuit player have to satisfy
1. if \( i \) is visited infinitely often, then \( s \) as well, and
2. \( ℓ \) is never visited.

Equivalently: color the vertices by natural numbers as above and require that almost all odd colors are followed by a larger even one.

This is the classical parity condition for \( \omega \) -automata.
Consider the one-client case!

Winning plays for circuit player have to satisfy

1. if i is visited infinitely often, then o as well, and
2. if o is visited, then it has not been visited since the last visit of i.

This requires an expressive specification language, e.g., Linear Temporal Logic (LTL).
Consider the one-client case!

\[ \ell \stackrel{i}{\longrightarrow} \bar{i} \stackrel{s}{\longrightarrow} i \]

1. If \( i \) is visited infinitely often, then \( s \) as well, and
2. \( \ell \) is never visited.

Equivalently:
- Color the vertices by natural numbers as above and
  - Require that almost all odd colors are followed by a larger even one.

This is the classical parity condition for \( \omega \)-automata.
Consider the one-client case!

Now, the winning plays for the circuit player have to satisfy:

1. if \( i \) is visited infinitely often, then \( s \) as well, and
2. \( l \) is never visited.

Equivalently:

- color the vertices by natural numbers as above and require that almost all odd colors are followed by a larger even one.

This is the classical parity condition for \( \omega \)-automata.
Consider the one-client case!

Now, the winning plays for the circuit player have to satisfy

1. if $i$ is visited infinitely often, then $s$ as well, and
2. $\ell$ is never visited.
Consider the one-client case!

Now, the winning plays for the circuit player have to satisfy

1. if $i$ is visited infinitely often, then $s$ as well, and
2. $\ell$ is never visited.

**Equivalently:** color the vertices by natural numbers as above and require that almost all odd colors are followed by a larger even one.

This is the classical **parity condition** for $\omega$-automata.
Player 0 has a memoryless winning strategy, which can be turned into an automaton with output, which can be turned into a circuit satisfying the specification.
Player 0 has a memoryless winning strategy,
Player 0 has a memoryless winning strategy,
which can be turned into an automaton with output,
Player 0 has a memoryless winning strategy,
which can be turned into an automaton with output,
which can be turned into a circuit satisfying the specification.
Quality of winning strategies is measured in multiple dimensions:

1. Memory requirements
2. Degree of satisfaction of a (quantitative) winning condition
3. Computational complexity of computing such a strategy
4. Use of lookahead
5. Use of randomization
6. Informedness
7. Robustness
8. Non-determinism

In previous work, each dimension was studied in isolation.
Quality of winning strategies is measured in multiple dimensions:

1. Memory requirements
2. Degree of satisfaction of a (quantitative) winning condition
3. Computational complexity of computing such a strategy
4. Use of lookahead
5. Use of randomization
6. Informedness
7. Robustness
8. Non-determinism

In previous work, each dimension was studied in isolation.

Goal of this thesis:
Understand the tradeoffs between (some of) these dimensions.
Overview

Typical questions we answered in this thesis:

- What is the price of optimality?
  - Is it harder to compute an optimal winning strategy in a quantitative game than an arbitrary winning strategy?
  - Does optimality increase the memory requirements of winning strategies?

Can the expressiveness of LTL be increased without increasing the complexity of solving games?

How does the addition of lookahead change the characteristics of games?

Does lookahead increase the complexity of solving games?

Does lookahead allow to improve the quality of strategies in quantitative games?
Overview

Typical questions we answered in this thesis:

- What is the price of optimality?
  - Is it harder to compute an optimal winning strategy in a quantitative game than an arbitrary winning strategy?
  - Does optimality increase the memory requirements of winning strategies?
- Can the expressiveness of LTL be increased without increasing the complexity of solving games?

How does the addition of lookahead change the characteristics of games?
- Does lookahead increase the complexity of solving games?
- Does lookahead allow to improve the quality of strategies in quantitative games?
Overview

Typical questions we answered in this thesis:

- What is the price of optimality?
  - Is it harder to compute an optimal winning strategy in a quantitative game than an arbitrary winning strategy?
  - Does optimality increase the memory requirements of winning strategies?
- Can the expressiveness of LTL be increased without increasing the complexity of solving games?
- How does the addition of lookahead change the characteristics of games?
  - Does lookahead increase the complexity of solving games?
  - Does lookahead allow to improve the quality of strategies in quantitative games?
Outline

1. Playing Optimally in Variations of Parity Games
2. Playing (Approximatively) Optimally in LTL Games
3. More Tradeoffs
   - Lookahead vs. Quality
   - Lookahead vs. Memory
   - Expressiveness vs. Complexity
   - Expressiveness vs. Memory
4. Conclusion
Outline

1. Playing Optimally in Variations of Parity Games
2. Playing (Approximatively) Optimally in LTL Games
3. More Tradeoffs
4. Conclusion
A Parity Game

Player 0 wins from every vertex, but Player 1 can delay between color 1 and color 2 longer and longer.

⇒ undesired behavior.
A Parity Game

Player 0 wins from every vertex, but Player 1 can delay between color 1 and color 2 longer and longer.

⇒ undesired behavior.

1 → 0 → 2

1 → 0 → 0 → 2

1 → 0 → 0 → 0 → 2 → ⋯
A Parity Game

Player 0 wins from every vertex, but Player 1 can delay between color 1 and color 2 longer and longer.

⇒ undesired behavior.

2 steps
1 → 0 → 2

3 steps
1 → 0 → 0 → 2

4 steps
1 → 0 → 0 → 0 → 2 → ...
A Parity Game

Player 0 wins from every vertex, but Player 1 can delay between color 1 and color 2 longer and longer. ⇒ undesired behavior.
During the last two decades, various quantitative variants of parity games have been introduced:

- Mean-payoff parity games [CHJ05]
- Finitary parity games [CH06]
- Energy parity games [CD11]
During the last two decades, various quantitative variants of parity games have been introduced:

- Mean-payoff parity games [CHJ05]
- Finitary parity games [CH06]
- Energy parity games [CD11]

Finitary parity games are distinguished, as here the quantitative aspect measures the satisfaction of the qualitative one.
Parity Games go Quantitative

During the last two decades, various quantitative variants of parity games have been introduced:

- Mean-payoff parity games [CHJ05]
- Finitary parity games [CH06]
- Energy parity games [CD11]
- Window parity games [BHR16]

Finitary parity games are distinguished, as here the quantitative aspect measures the satisfaction of the qualitative one.
**Finitary Parity Games**

- **Parity:** Almost all odd colors are followed by larger even one.
- **Finitary Parity:** There is a bound $b$ such that almost all odd colors are followed by larger even one within $b$ steps.
Parity: Almost all odd colors are followed by larger even one.

Finitary Parity: There is a bound $b$ such that almost all odd colors are followed by larger even one within $b$ steps.

<table>
<thead>
<tr>
<th>Condition</th>
<th>Complexity</th>
<th>Memory Pl. 0</th>
<th>Memory Pl. 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parity</td>
<td>quasi-poly</td>
<td>Memoryless</td>
<td>Memoryless</td>
</tr>
<tr>
<td>Finitary Parity</td>
<td>PTIME</td>
<td>Memoryless</td>
<td>Infinite</td>
</tr>
</tbody>
</table>
The bound $b$ in the definition of finitary parity games is existentially quantified (and may depend on the play).

**Corollary**

*If Player 0 wins a finitary parity game $G$, then a uniform bound $b \leq |G|$ suffices.*

A trivial example shows that the upper bound $|G|$ is tight.
The bound $b$ in the definition of finitary parity games is existentially quantified (and may depend on the play).

**Corollary**

*If Player 0 wins a finitary parity game $\mathcal{G}$, then a uniform bound $b \leq |\mathcal{G}|$ suffices.*

A trivial example shows that the upper bound $|\mathcal{G}|$ is tight.

**Questions**

1. Does Player 0 need memory to achieve the optimal bound?
2. Is it harder to compute the optimal bound than checking whether a bound exists?
Player 0 has a unique memoryless winning strategy, which achieves the bound five: from the 1 it takes five steps to the 4. With two memory states, she can achieve the bound four: "answer" a 1 by a 2 and a 3 by a 4. It is trivial to extend this example to $d$ odd colors and $d$ even colors requiring $d$ memory states to play optimally.

⇒ In general, playing optimally requires memory; but how much?
Player 0 has a unique memoryless winning strategy, which achieves the bound five: from the 1 it takes five steps to the 4.
Player 0 has a unique memoryless winning strategy, which achieves the bound five: from the 1 it takes five steps to the 4.

With two memory states, she can achieve the bound four: “answer” a 1 by a 2 and a 3 by a 4.
Player 0 has a unique memoryless winning strategy, which achieves the bound five: from the 1 it takes five steps to the 4.

With two memory states, she can achieve the bound four: “answer” a 1 by a 2 and a 3 by a 4.

It is trivial to extend this example to $d$ odd colors and $d$ even colors requiring $d$ memory states to play optimally.

⇒ In general, playing optimally requires memory; but how much?
Memory Requirements

\[
\begin{array}{cccc}
0 & 0 & \cdots & 0 \\
1 & 3 & \cdots & 2d - 1 \\
0 & 0 & \cdots & 0 \\
\end{array}
\]

\[
\begin{array}{cccc}
0 & 0 & \cdots & 0 \\
2 & 4 & \cdots & 2d \\
0 & 0 & \cdots & 0 \\
\end{array}
\]
Memory Requirements

- $d$ request gadgets with $d$ colors
- $d$ response gadgets with $d$ colors

Player 0 has winning strategy with cost $d^2 + 2d$: answer $j$-th unique request in $j$-th response-gadget.\[\Rightarrow\text{requires exponential memory (in } d)\.

Against a smaller strategy Player 1 can enforce a larger cost, as Player 0 cannot store every sequence of requests.

Theorem (WZ16)
For every $d > 1$, there exists a finitary parity game $G_d$ such that $|G_d| \in O(d^2)$ and $G_d$ has $d$ odd colors, and every optimal strategy for Player 0 has at least size $2^{d-1}$.
Player 0 has winning strategy with cost $d^2 + 2d$: answer $j$-th unique request in $j$-th response-gadget.

⇒ requires exponential memory (in $d$).
Player 0 has winning strategy with cost $d^2 + 2d$: answer $j$-th unique request in $j$-th response-gadget. 
$\Rightarrow$ requires exponential memory (in $d$).
Against a smaller strategy Player 1 can enforce a larger cost, as Player 0 cannot store every sequence of requests.
Memory Requirements

- Player 0 has winning strategy with cost $d^2 + 2d$: answer $j$-th unique request in $j$-th response-gadget.
  \[ \Rightarrow \text{requires exponential memory (in } d) \]

- Against a smaller strategy Player 1 can enforce a larger cost, as Player 0 cannot store every sequence of requests.

**Theorem (WZ16)**

*For every $d > 1$, there exists a finitary parity game $G_d$ such that*

- $|G_d| \in \mathcal{O}(d^2)$ and $G_d$ has $d$ odd colors, and
- every optimal strategy for Player 0 has at least size $2^{d-1}$. 
Lemma (WZ16)

The following problem is PSPACE-hard: “Given a finitary parity game $\mathcal{G}$ and a bound $b \in \mathbb{N}$, does Player 0 have a strategy for $\mathcal{G}$ whose cost is at most $b$?”
Lemma (WZ16)

The following problem is PSPACE-hard: “Given a finitary parity game $G$ and a bound $b \in \mathbb{N}$, does Player 0 have a strategy for $G$ whose cost is at most $b$?”

Proof

- By a reduction from QBF (w.l.o.g. in CNF).
- Checking the truth of $\varphi = \forall x \exists y. (x \lor \neg y) \land (\neg x \lor y)$ as a two-player game (Player 0 wants to prove truth of $\varphi$):
Lemma (WZ16)

The following problem is $\mathsf{PSPACE}$-hard: “Given a finitary parity game $\mathcal{G}$ and a bound $b \in \mathbb{N}$, does Player 0 have a strategy for $\mathcal{G}$ whose cost is at most $b$?”

Proof

- By a reduction from QBF (w.l.o.g. in CNF).
- Checking the truth of $\varphi = \forall x \exists y. (x \lor \neg y) \land (\neg x \lor y)$ as a two-player game (Player 0 wants to prove truth of $\varphi$):
  1. Player 1 picks truth value for $x$.
  2. Player 0 picks truth value for $y$.
  3. Player 1 picks clause $C$.
  4. Player 0 picks literal $\ell$ from $C$.
  5. Player 0 wins $\iff \ell$ is picked to be satisfied in step 1 or 2.
The Reduction

\[ \varphi = \forall x \ \exists y . \ \left( x \lor \neg y \right) \land \left( \neg x \lor y \right) \]
The Reduction

\[ \varphi = \forall x \exists y. \left( (x \lor \neg y) \land (\neg x \lor y) \right) \]
The Reduction

\[ \varphi = \forall x \exists y . \left( x \lor \neg y \right) \land \left( \neg x \lor y \right) \]

For a well-chosen bound \( b \), a strategy for Player 0 with cost at most \( b \) witnesses the truth of \( \varphi \) and vice versa.
The Reduction

\[ \varphi = \forall x . \exists y . (x \lor \neg y) \land (\neg x \lor y) \]

For a well-chosen bound \( b \), a strategy for Player 0 with cost at most \( b \) witnesses the truth of \( \varphi \) and vice versa.
The Reduction

\[ \varphi = \forall x \ \exists y . \ (x \lor \neg y) \land (\neg x \lor y) \]

For a well-chosen bound \( b \), a strategy for Player 0 with cost at most \( b \) witnesses the truth of \( \varphi \) and vice versa.
The Reduction

\[
\varphi = \forall x \ \exists y . \left( x \lor \neg y \right) \land \left( \neg x \lor y \right)
\]

For a well-chosen bound \( b \), a strategy for Player 0 with cost at most \( b \) witnesses the truth of \( \varphi \) and vice versa.
The Reduction

\[ \varphi = \forall x \, \exists y . \left( x \lor \neg y \right) \land \left( \neg x \lor y \right) \]

For a well-chosen bound \( b \), a strategy for Player 0 with cost at most \( b \) witnesses the truth of \( \varphi \) and vice versa.
The Reduction

\[ \varphi = \forall x \, \exists y \cdot (x \lor \neg y) \land (\neg x \lor y) \]

For a well-chosen bound \( b \), a strategy for Player 0 with cost at most \( b \) witnesses the truth of \( \varphi \) and vice versa.
The Reduction

\[ \varphi = \forall x \; \exists y \cdot (x \lor \neg y) \land (\neg x \lor y) \]

For a well-chosen bound \( b \), a strategy for Player 0 with cost at most \( b \) witnesses the truth of \( \varphi \) and vice versa.
The Reduction

\[ \varphi = \forall x \, \exists y \cdot (x \lor \neg y) \land (\neg x \lor y) \]

For a well-chosen bound \( b \), a strategy for Player 0 with cost at most \( b \) witnesses the truth of \( \varphi \) and vice versa.
The Reduction

\[ \varphi = \forall x \exists y . \ (x \lor \neg y) \land (\neg x \lor y) \]

For a well-chosen bound \( b \), a strategy for Player 0 with cost at most \( b \) witnesses the truth of \( \varphi \) and vice versa.
The Reduction

$$\varphi = \forall x \ \exists y \ . \ (x \lor \neg y) \land (\neg x \lor y)$$

For a well-chosen bound $b$, a strategy for Player 0 with cost at most $b$ witnesses the truth of $\varphi$ and vice versa.
The Reduction

\[ \varphi = \forall x \, \exists y \cdot (x \lor \neg y) \land (\neg x \lor y) \]

For a well-chosen bound \( b \), a strategy for Player 0 with cost at most \( b \) witnesses the truth of \( \varphi \) and vice versa.
The Reduction

\[ \varphi = \forall x \ \exists y . \ \left( (x \lor \neg y) \land (\neg x \lor y) \right) \]

For a well-chosen bound \( b \), a strategy for Player 0 with cost at most \( b \) witnesses the truth of \( \varphi \) and vice versa.
The Reduction

\[ \varphi = \forall x \ \exists y . \ (x \lor \neg y) \land (\neg x \lor y) \]

For a well-chosen bound \( b \), a strategy for Player 0 with cost at most \( b \) witnesses the truth of \( \varphi \) and vice versa.
The Reduction

\[ \varphi = \forall x \ \exists y . \ (x \lor \neg y) \land (\neg x \lor y) \]

For a well-chosen bound \( b \), a strategy for Player 0 with cost at most \( b \) witnesses the truth of \( \varphi \) and vice versa.
Lemma (WZ16)

The following problem is in $\text{PSPACE}$: “Given a finitary parity game $G$ and a bound $b \in \mathbb{N}$, does Player 0 have a strategy for $G$ whose cost is at most $b$?”
Lemma (WZ16)

The following problem is in $\text{PSPACE}$: “Given a finitary parity game $G$ and a bound $b \in \mathbb{N}$, does Player 0 have a strategy for $G$ whose cost is at most $b$?”

Proof Sketch

Fix $G$ and $b$ (w.l.o.g. $b \leq |G|$).

1. Construct equivalent parity game $G'$ storing the costs of open requests (up to bound $b$) and the number of “overflows” (up to bound $|G|$) $\Rightarrow |G'| \in |G|^{O(d)}$. 
Lemma (WZ16)

The following problem is in \( \text{PSPACE} \): “Given a finitary parity game \( G \) and a bound \( b \in \mathbb{N} \), does Player 0 have a strategy for \( G \) whose cost is at most \( b \)?”

Proof Sketch

Fix \( G \) and \( b \) (w.l.o.g. \( b \leq |G| \)).

1. Construct equivalent parity game \( G' \) storing the costs of open requests (up to bound \( b \)) and the number of “overflows” (up to bound \( |G| \)) \( |G'| \in O(d) \).

2. Define equivalent finite-duration variant \( G'_f \) of \( G' \) with polynomial play-length.

3. \( G'_f \) can be solved on alternating polynomial-time Turing machine.

4. \( \text{APTime} = \text{PSPACE} \) concludes the proof.
Equivalence between finitary parity game $G$ w.r.t. bound $b$ and parity game $G'$ yields upper bounds on memory requirements.

**Corollary**

Let $G$ be a finitary parity game with costs with $d$ odd colors. If Player 0 has a strategy for $G$ with cost $b$, then she also has a strategy with cost $b$ and size $(b + 2)^d = 2^d \log(b+2)$. Recall: lower bound $2^d - 1$. The same bounds hold for Player 1.
Equivalence between finitary parity game $\mathcal{G}$ w.r.t. bound $b$ and parity game $\mathcal{G}'$ yields upper bounds on memory requirements.

**Corollary**

Let $\mathcal{G}$ be a finitary parity game with costs with $d$ odd colors. If Player 0 has a strategy for $\mathcal{G}$ with cost $b$, then she also has a strategy with cost $b$ and size $(b + 2)^d = 2^d \log(b+2)$.

- Recall: lower bound $2^{d-1}$.
- The same bounds hold for Player 1.
Theorem (WZ16)

Fix some finitary parity game $G_d$ as before. For every $i$ with $1 \leq i \leq d$ there exists a strategy $\sigma_i$ for Player 0 in $G_d$ such that $\sigma_i$ has cost $d^2 + 3d - i$ and size $\sum_{j=1}^{i-1} (d_j)$. Also, every strategy $\sigma'$ for Player 0 in $G_d$ whose cost is at most the cost of $\sigma_i$ has at least the size of $\sigma_i$. 
We generalized finitary parity games to
Generalizations

We generalized finitary parity games to

- parity games with costs (by allowing non-negative weights on the edges) \([\text{FZ14}]\), and
We generalized finitary parity games to
- parity games with costs (by allowing non-negative weights on the edges) [FZ14], and
- parity games with weights (by allowing arbitrary weights on the edges) [SWZ18].
Generalizations

We generalized finitary parity games to
- parity games with costs (by allowing non-negative weights on the edges) [FZ14], and
- parity games with weights (by allowing arbitrary weights on the edges) [SWZ18].

Boundedness

<table>
<thead>
<tr>
<th>Condition</th>
<th>Complexity</th>
<th>Memory Pl. 0</th>
<th>Memory Pl. 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parity</td>
<td>quasi-poly</td>
<td>Memoryless</td>
<td>Memoryless</td>
</tr>
<tr>
<td>Finitary Parity</td>
<td>PTIME</td>
<td>Memoryless</td>
<td>Infinite</td>
</tr>
<tr>
<td>Parity w. Costs</td>
<td>quasi-poly</td>
<td>Memoryless</td>
<td>Infinite</td>
</tr>
<tr>
<td>Parity w. Weights</td>
<td>NP ∩ co-NP</td>
<td>Exponential</td>
<td>Infinite</td>
</tr>
</tbody>
</table>
Generalizations

We generalized finitary parity games to

- parity games with costs (by allowing non-negative weights on the edges) \([FZ14]\), and
- parity games with weights (by allowing arbitrary weights on the edges) \([SWZ18]\).

Optimization

<table>
<thead>
<tr>
<th>Condition</th>
<th>Complexity</th>
<th>Memory Pl. 0 &amp; 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finitary Parity</td>
<td>(PSPACE)-complete</td>
<td>Exponential</td>
</tr>
<tr>
<td>Parity w. Costs</td>
<td>(PSPACE)-complete</td>
<td>Exponential</td>
</tr>
<tr>
<td>Parity w. Weights</td>
<td>(PSPACE)-hard</td>
<td>(\geq) Exponential</td>
</tr>
</tbody>
</table>

- The results for parity games with costs hold for unary and binary encodings of the weights.
Outline

1. Playing Optimally in Variations of Parity Games
2. Playing (Approximatively) Optimally in LTL Games
3. More Tradeoffs
4. Conclusion
Atomic propositions $r_i$ for requests and $g_i$ for grants.

1. Answer every request: $\bigwedge_i \mathbf{G} (r_i \rightarrow \mathbf{F} g_i)$
Atomic propositions $r_i$ for requests and $g_i$ for grants.

1. Answer every request: $\bigwedge_i G (r_i \rightarrow F g_i)$

2. At most one grant at a time: $G \bigwedge_{i \neq j} \neg(g_i \land g_j)$
Atomic propositions $r_i$ for requests and $g_i$ for grants.

1. Answer every request: $\bigwedge_i \mathsf{G}(r_i \rightarrow \mathsf{F} g_i)$

2. At most one grant at a time: $\mathsf{G} \bigwedge_{i \neq j} \neg (g_i \land g_j)$

3. No spurious grants:

$$\bigwedge_i \neg [ (\neg r_i \mathsf{U} (\neg r_i \land g_i)) ] \land \neg [ \mathsf{F}(g_i \land \mathsf{X}(\neg r_i \mathsf{U} (\neg r_i \land g_i))) ]$$
Atomic propositions $r_i$ for requests and $g_i$ for grants.

1. Answer every request: $\bigwedge_i G (r_i \rightarrow F g_i)$

2. At most one grant at a time: $G \bigwedge_{i \neq j} \neg (g_i \land g_j)$

3. No spurious grants:

\[
\bigwedge_i \neg [(\neg r_i \mathbf{U} (\neg r_i \land g_i))] \land \neg [F (g_i \land X (\neg r_i \mathbf{U} (\neg r_i \land g_i)))]
\]

\[
\equiv \bigwedge_i [(r_i \mathbf{R} (r_i \lor \neg g_i))] \land [G (\neg g_i \lor X (r_i \mathbf{R} (r_i \lor \neg g_i)))]
\]
A Problem with LTL

Answer every request: \[ \bigwedge_i \mathbf{G}(r_i \rightarrow \mathbf{F} g_i) \]

\[
\begin{align*}
r_0 & \rightarrow g_0 \rightarrow r_0 \rightarrow g_0 \rightarrow r_0 \rightarrow g_0 \rightarrow \cdots
\end{align*}
\]
A Problem with LTL

Answer every request: $\bigwedge_i \mathbf{G}(r_i \rightarrow F g_i)$

$r_0 \rightarrow g_0 \rightarrow r_0 \rightarrow \cdots g_0 \rightarrow r_0 \rightarrow \cdots g_0 \rightarrow \cdots$

Problem:
LTL is too weak to express timing-constraints: no guarantee when request is granted, only that it is granted eventually
LTL goes Quantitative

During the last two decades, various quantitative variants of LTL have been introduced:

- Parametric LTL \([\text{AETP99}]\)
- PROMPT–LTL \([\text{KPV07}]\)
- Parametric MTL \([\text{GTN10}]\)
During the last two decades, various quantitative variants of LTL have been introduced:

- Parametric LTL [AETP99]
- PROMPT–LTL [KPV07]
- Parametric MTL [GTN10]

PROMPT–LTL is distinguished, as all problems for the more general Parametric LTL are reducible to those for PROMPT–LTL.
Prompt-LTL


$$\varphi ::= p \mid \neg p \mid \varphi \land \varphi \mid \varphi \lor \varphi \mid X \varphi \mid \varphi U \varphi \mid \varphi R \varphi \mid F_p \varphi$$
Prompt-LTL

**Syntax:** Add prompt-eventually operator $F_P$.

$$\varphi ::= p \mid \neg p \mid \varphi \land \varphi \mid \varphi \lor \varphi \mid X \varphi \mid \varphi U \varphi \mid \varphi R \varphi \mid F_P \varphi$$

**Semantics:** Defined with respect to a fixed bound $k \in \mathbb{N}$.

$$\models (\rho, n, k) \models F_P \varphi: \quad \rho \ldots n \quad \varphi \quad n + k$$
Prompt-LTL

**Syntax:** Add prompt-eventually operator $F_P$.

$$\phi ::= p \mid \neg p \mid \phi \land \phi \mid \phi \lor \phi \mid X \phi \mid \phi U \phi \mid \phi R \phi \mid F_P \phi$$

**Semantics:** Defined with respect to a fixed bound $k \in \mathbb{N}$.

$$\models (\rho, n, k) \models F_P \phi: \quad \rho \cdots n \quad \downarrow \quad \phi \quad n + k$$

Now: $\bigwedge_i G (x_i \rightarrow F_P g_i)$
Label the arena by atomic propositions.

Winning condition: PROMPT–LTL formula $\varphi$.

Player 0 wins if there is a uniform bound $k$ and a strategy $\sigma$ such that every play that is consistent with $\sigma$ satisfies the winning condition $\varphi$ w.r.t. $k$. 
Prompt-LTL Games

- Label the arena by atomic propositions.
- Winning condition: PROMPT–LTL formula $\varphi$.
- Player 0 wins if there is a uniform bound $k$ and a strategy $\sigma$ such that every play that is consistent with $\sigma$ satisfies the winning condition $\varphi$ w.r.t. $k$.

PROMPT–LTL games are not harder than LTL games...

Theorem (KPV07)

1. Determining the winner of PROMPT–LTL games is $2\text{ExpTime}$-complete.
2. If Player 0 wins, then also with a finite-state strategy of size $2^{2|\varphi|}$ and w.r.t. the bound $k_\varphi = 2^{2|\varphi|}$.
Label the arena by atomic propositions.

Winning condition: PROMPT–LTL formula $\varphi$.

Player 0 wins if there is a uniform bound $k$ and a strategy $\sigma$ such that every play that is consistent with $\sigma$ satisfies the winning condition $\varphi$ w.r.t. $k$.

...unless you optimize the bound.

**Theorem (Z11)**

1. The PROMPT–LTL game optimization problem can be solved in triply-exponential time.

2. The bound $k_\varphi$ is tight in general.
Prompt-LTL Games

- Label the arena by atomic propositions.
- Winning condition: PROMPT–LTL formula $\varphi$.
- Player 0 wins if there is a uniform bound $k$ and a strategy $\sigma$ such that every play that is consistent with $\sigma$ satisfies the winning condition $\varphi$ w.r.t. $k$.

Questions

1. Is the optimization problem harder than the boundedness problem?
2. Can the optimum be approximated?
An Approximation Algorithm

Lemma (TWZ16)

Fix a PROMPT–LTL game $G$ with winning condition $\varphi$ and $k \leq k_\varphi$. There is an LTL game $G_k$ such that

1. if Player 0 wins $G$ w.r.t. $k$, then she wins $G_k$,
2. if Player 0 wins $G_k$, then she wins $G$ w.r.t. $2k$, and
3. $G_k$ can be solved in doubly-exponential time in $|G|$.
**An Approximation Algorithm**

**Lemma (TWZ16)**

Fix a PROMPT–LTL game \( G \) with winning condition \( \varphi \) and \( k \leq k_\varphi \). There is an LTL game \( G_k \) such that

1. if Player 0 wins \( G \) w.r.t. \( k \), then she wins \( G_k \),
2. if Player 0 wins \( G_k \), then she wins \( G \) w.r.t. \( 2k \), and
3. \( G_k \) can be solved in doubly-exponential time in \( |G| \).

The algorithm:

1: **for** \( k = 0; \; k \leq k_\varphi; \; k \leftarrow k + 1 \) **do**
2: **if** Player 0 wins \( G_k \) **then**
3: **return** \( 2k \)
An Approximation Algorithm

Lemma (TWZ16)

Fix a PROMPT–LTL game $G$ with winning condition $\varphi$ and $k \leq k_\varphi$. There is an LTL game $G_k$ such that

1. if Player 0 wins $G$ w.r.t. $k$, then she wins $G_k$,
2. if Player 0 wins $G_k$, then she wins $G$ w.r.t. $2k$, and
3. $G_k$ can be solved in doubly-exponential time in $|G|$.

The algorithm:

1: for $k = 0; k \leq k_\varphi; k \leftarrow k + 1$ do
2: if Player 0 wins $G_k$ then
3: return $2k$

- Running time: doubly-exponential
- Approximation ratio: 2
- Yields winning strategy
Implementation via bounded synthesis:

- Search for finite-state strategy of size $n$ achieving bound $k$.
- Complete due to upper bounds on $n$ and $k$. 
Implementation via **bounded synthesis**:

- Search for finite-state strategy of size $n$ achieving bound $k$.
- Complete due to upper bounds on $n$ and $k$. 
Implementation via **bounded synthesis**:  
- Search for finite-state strategy of size $n$ achieving bound $k$.  
- Complete due to upper bounds on $n$ and $k$.  

---

**Tradeoffs**
Implementation via bounded synthesis:

- Search for finite-state strategy of size $n$ achieving bound $k$.
- Complete due to upper bounds on $n$ and $k$. 
Implementation via \textit{bounded synthesis}:

- Search for finite-state strategy of size $n$ achieving bound $k$.
- Complete due to upper bounds on $n$ and $k$. 
Implementation via bounded synthesis:

- Search for finite-state strategy of size $n$ achieving bound $k$.
- Complete due to upper bounds on $n$ and $k$. 
Implementation via **bounded synthesis**:

- Search for finite-state strategy of size $n$ achieving bound $k$.
- Complete due to upper bounds on $n$ and $k$. 
Implementation via **bounded synthesis**:

- Search for finite-state strategy of size $n$ achieving bound $k$.
- Complete due to upper bounds on $n$ and $k$. 
Implementation via **bounded synthesis**:

- Search for finite-state strategy of size $n$ achieving bound $k$.
- Complete due to upper bounds on $n$ and $k$. 
Implementation via bounded synthesis:

- Search for finite-state strategy of size \( n \) achieving bound \( k \).
- Complete due to upper bounds on \( n \) and \( k \).
Implementation via **bounded synthesis**:

- Search for finite-state strategy of size $n$ achieving bound $k$.
- Complete due to upper bounds on $n$ and $k$. 

---

Martin Zimmermann  
Saarland University  
Tradeoffs in Infinite Games
Implementation via bounded synthesis:

- Search for finite-state strategy of size $n$ achieving bound $k$.
- Complete due to upper bounds on $n$ and $k$. 
Tradeoffs

Implementation via **bounded synthesis**:

- Search for finite-state strategy of size $n$ achieving bound $k$.
- Complete due to upper bounds on $n$ and $k$. 

Diagram:

- Vertical axis labeled "bound"
- Horizontal axis labeled "memory size"
- Filled grid indicating search space
Implementation via **bounded synthesis**:

- Search for finite-state strategy of size $n$ achieving bound $k$.
- Complete due to upper bounds on $n$ and $k$. 
Implementation via bounded synthesis:

- Search for finite-state strategy of size $n$ achieving bound $k$.
- Complete due to upper bounds on $n$ and $k$. 
Implementation via \textit{bounded synthesis}:

- Search for finite-state strategy of size $n$ achieving bound $k$.
- Complete due to upper bounds on $n$ and $k$. 
Implementation via **bounded synthesis:**

- Search for finite-state strategy of size $n$ achieving bound $k$.
- Complete due to upper bounds on $n$ and $k$. 
Implementation via **bounded synthesis:**

- Search for finite-state strategy of size $n$ achieving bound $k$.
- Complete due to upper bounds on $n$ and $k$. 

Tradeoffs
Implementation via **bounded synthesis**: 

- Search for finite-state strategy of size $n$ achieving bound $k$. 
- Complete due to upper bounds on $n$ and $k$. 
Tradeoffs

Implementation via bounded synthesis:

- Search for finite-state strategy of size $n$ achieving bound $k$.
- Complete due to upper bounds on $n$ and $k$. 
Implementation via bounded synthesis:
- Search for finite-state strategy of size $n$ achieving bound $k$.
- Complete due to upper bounds on $n$ and $k$. 
Implementation via bounded synthesis:

- Search for finite-state strategy of size $n$ achieving bound $k$.
- Complete due to upper bounds on $n$ and $k$.
- $\mathcal{R}_G$: realizable combinations for game $G$. 
Implementation via bounded synthesis:
- Search for finite-state strategy of size $n$ achieving bound $k$.
- Complete due to upper bounds on $n$ and $k$.
- $\mathcal{R}_G$: realizable combinations for game $G$. 
Tradeoffs

Implementation via bounded synthesis:
- Search for finite-state strategy of size $n$ achieving bound $k$.
- Complete due to upper bounds on $n$ and $k$.
- $R_G$: realizable combinations for game $G$. 
Implementation via **bounded synthesis**:

- Search for finite-state strategy of size \( n \) achieving bound \( k \).
- Complete due to upper bounds on \( n \) and \( k \).
- \( \mathcal{R}_G \): realizable combinations for game \( G \).
Implementation via bounded synthesis:

- Search for finite-state strategy of size $n$ achieving bound $k$.
- Complete due to upper bounds on $n$ and $k$.
- $\mathcal{R}_G$: realizable combinations for game $G$. 
Implementation via bounded synthesis:

- Search for finite-state strategy of size $n$ achieving bound $k$.
- Complete due to upper bounds on $n$ and $k$.
- $\mathcal{R}_G$: realizable combinations for game $\mathcal{G}$.
Tradeoffs in Infinite Games

Implementation via **bounded synthesis**:

- Search for finite-state strategy of size \( n \) achieving bound \( k \).
- Complete due to upper bounds on \( n \) and \( k \).
- \( \mathcal{R}_G \): realizable combinations for game \( G \).
Implementation via bounded synthesis:

- Search for finite-state strategy of size $n$ achieving bound $k$.
- Complete due to upper bounds on $n$ and $k$.
- $\mathcal{R}_G$: realizable combinations for game $\mathcal{G}$. 
Implementation via **bounded synthesis**:

- Search for finite-state strategy of size $n$ achieving bound $k$.
- Complete due to upper bounds on $n$ and $k$.
- $\mathcal{R}_G$: realizable combinations for game $G$. 
Implementation via **bounded synthesis**:  
- Search for finite-state strategy of size $n$ achieving bound $k$.  
- Complete due to upper bounds on $n$ and $k$.  
- $\mathcal{R}_G$: realizable combinations for game $\mathcal{G}$.  

---  

Martin Zimmermann  
Saarland University  
Tradeoffs in Infinite Games  
29/36
Implementation via \textit{bounded synthesis}:

- Search for finite-state strategy of size $n$ achieving bound $k$.
- Complete due to upper bounds on $n$ and $k$.
- $\mathcal{R}_G$: realizable combinations for game $G$. 
Implementation via bounded synthesis:

- Search for finite-state strategy of size $n$ achieving bound $k$.
- Complete due to upper bounds on $n$ and $k$.
- $\mathcal{R}_G$: realizable combinations for game $G$. 
Implementation via **bounded synthesis:**

- Search for finite-state strategy of size $n$ achieving bound $k$.
- Complete due to upper bounds on $n$ and $k$.
- $\mathcal{R}_G$: realizable combinations for game $\mathcal{G}$. 
Implementation via \textit{bounded synthesis}:

- Search for finite-state strategy of size $n$ achieving bound $k$.
- Complete due to upper bounds on $n$ and $k$.
- $\mathcal{R}_G$: realizable combinations for game $G$. 
Implementation via **bounded synthesis**:

- Search for finite-state strategy of size $n$ achieving bound $k$.
- Complete due to upper bounds on $n$ and $k$.
- $\mathcal{R}_G$: realizable combinations for game $G$. 

---

Tradeoffs in Infinite Games
Implementation via **bounded synthesis**:  
- Search for finite-state strategy of size $n$ achieving bound $k$.  
- Complete due to upper bounds on $n$ and $k$.  
- $\mathcal{R}_G$: realizable combinations for game $\mathcal{G}$.  

**Tradeoffs**
Implementation via bounded synthesis:

- Search for finite-state strategy of size $n$ achieving bound $k$.
- Complete due to upper bounds on $n$ and $k$.
- $\mathcal{R}_G$: realizable combinations for game $G$. 

\section*{Tradeoffs}
Implementation via bounded synthesis:

- Search for finite-state strategy of size $n$ achieving bound $k$.
- Complete due to upper bounds on $n$ and $k$.
- $\mathcal{R}_G$: realizable combinations for game $G$. 
Lemma

1. If \((n, k) \in R_G\), then \((n, n^{2^{|\varphi|}}) \in R_G\).

2. If \((n, k) \in R_G\), then \((2^{2^k|\varphi|}, k) \in R_G\).
Theorem (TWZ16)

For every $b$, there is a game $\mathcal{G}_b$ of size $\mathcal{O}(b)$ such that $(2^j, 2^{b-j})$ is a Pareto point for every $j \leq b$. 

\[\mathcal{R}_{\mathcal{G}_b}\]
Theorem (TWZ16)

For every $b$, there is a game $\mathcal{G}_b$ of size $O(b)$ such that $(2^{2^b}, 0)$ and $(c, 2^b)$ are Pareto points for some constant $c$. 

bound

memory size

$R_{G_b}$

$c$

$2^b$

$2^{2^b}$
An Example

Five clients:

1. Answer every request of client 1 promptly: $\text{G} (r_1 \rightarrow \text{F}_p g_1)$
2. Answer every other request eventually: $\bigwedge_{i > 1} \text{G} (r_i \rightarrow \text{F} g_i)$
3. At most one grant at a time: $\text{G} \bigwedge_{i \neq j} \neg (g_i \land g_j)$
An Example

Five clients:

1. Answer every request of client 1 promptly: $G(r_1 \rightarrow F_p g_1)$
2. Answer every other request eventually: $\bigwedge_{i>1} G(r_i \rightarrow F g_i)$
3. At most one grant at a time: $G \bigwedge_{i \neq j} \neg(g_i \wedge g_j)$
An Example

Five clients:

1. Answer every request of client 1 promptly: $\mathbf{G}(r_1 \rightarrow F_P g_1)$
2. Answer every other request eventually: $\bigwedge_{i>1} \mathbf{G}(r_i \rightarrow F g_i)$
3. At most one grant at a time: $\mathbf{G} \bigwedge_{i \neq j} \neg (g_i \land g_j)$
Generalizations

- LDL with costs
- PLTL
- LTL
Generalizations

- Parametric LDL [FZ14]: full expressive power of the $\omega$-regular languages and bounded operators.
Generalizations

- Parametric LDL [FZ14]: full expressive power of the \( \omega \)-regular languages and bounded operators.
- Parametric LTL and LDL with costs [Z15]: replace unit cost by non-negative weights.
Generalizations

- Parametric LDL [FZ14]: full expressive power of the $\omega$-regular languages and bounded operators.
- Parametric LTL and LDL with costs [Z15]: replace unit cost by non-negative weights.

<table>
<thead>
<tr>
<th>Logic</th>
<th>Complexity</th>
<th>Memory Pl. 0 &amp; 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>LTL</td>
<td>$2\text{EXPTime}$-complete</td>
<td>Doubly-exponential</td>
</tr>
<tr>
<td>LDL</td>
<td>$2\text{EXPTime}$-complete</td>
<td>Doubly-exponential</td>
</tr>
<tr>
<td>PLTL</td>
<td>$2\text{EXPTime}$-complete</td>
<td>Doubly-exponential</td>
</tr>
<tr>
<td>PLDL</td>
<td>$2\text{EXPTime}$-complete</td>
<td>Doubly-exponential</td>
</tr>
<tr>
<td>cPLTL</td>
<td>$2\text{EXPTime}$-complete</td>
<td>Doubly-exponential</td>
</tr>
<tr>
<td>cPLDL</td>
<td>$2\text{EXPTime}$-complete</td>
<td>Doubly-exponential</td>
</tr>
</tbody>
</table>
Generalizations

- Parametric LDL [FZ14]: full expressive power of the $\omega$-regular languages and bounded operators.
- Parametric LTL and LDL with costs [Z15]: replace unit cost by non-negative weights.
- Visibly LDL [WZ15]: full expressive power of the $\omega$-visibly pushdown languages.

<table>
<thead>
<tr>
<th>Logic</th>
<th>Complexity</th>
<th>Memory Pl. 0 &amp; 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>LTL</td>
<td>$2^{\text{ExpTime}}$-complete</td>
<td>Doubly-exponential</td>
</tr>
<tr>
<td>LDL</td>
<td>$2^{\text{ExpTime}}$-complete</td>
<td>Doubly-exponential</td>
</tr>
<tr>
<td>PLTL</td>
<td>$2^{\text{ExpTime}}$-complete</td>
<td>Doubly-exponential</td>
</tr>
<tr>
<td>PLDL</td>
<td>$2^{\text{ExpTime}}$-complete</td>
<td>Doubly-exponential</td>
</tr>
<tr>
<td>cPLTL</td>
<td>$2^{\text{ExpTime}}$-complete</td>
<td>Doubly-exponential</td>
</tr>
<tr>
<td>cPLDL</td>
<td>$2^{\text{ExpTime}}$-complete</td>
<td>Doubly-exponential</td>
</tr>
<tr>
<td>VLDL</td>
<td>$3^{\text{ExpTime}}$-complete</td>
<td>pushdown transducer</td>
</tr>
</tbody>
</table>
1. Playing Optimally in Variations of Parity Games
2. Playing (Approximatively) Optimally in LTL Games
3. More Tradeoffs
4. Conclusion
Delay Games

- **Landweber & Hosch**: Allow one player to delay her moves to obtain a lookahead on the opponent’s moves.
- This thesis presents the first in-depth study of delay games.
Delay Games

- **Landweber & Hosch**: Allow one player to delay her moves to obtain a lookahead on the opponent’s moves.
- This thesis presents the first in-depth study of delay games.

**Theorem (KZ15)**

Solving $\omega$-regular delay games is $\text{ExpTime}$-complete and exponential lookahead is always sufficient and in general necessary.

Best previous result: in $2\text{ExpTime}$ and doubly-exponential upper bound, no non-trivial lower bounds [HKT10].
Delay Games

- **Landweber & Hosch**: Allow one player to delay her moves to obtain a lookahead on the opponent’s moves.
- This thesis presents the first in-depth study of delay games.

**More results** (incomplete)

- Solving LTL delay games is $3\text{ExPTIME}$-complete, triply-exponential lookahead sufficient and necessary \[ \text{KZ16}. \]
- Lookahead can be traded for quality in delay games with finitary parity conditions \[ \text{Z17}. \]
- Finite-state strategies for delay games \[ \text{WZ18}: \] lookahead can be traded for memory.
More Results

In the thesis, but not covered in this talk:

- Solving an open problem on average-energy games \[BHMRZ17\]
- Optimal strategies for request-response games \[HTWZ15\]
- Distributed synthesis for PROMPT–LTL \[JTZ16\]
- A first-order logic for Hyperproperties \[FZ17\]
- Context-free delay games \[FLZ11\]
- Borel determinacy for delay games \[KZ15\]
- Delay games with WMSO+U winning conditions \[Z15\]
Outline

1. Playing Optimally in Variations of Parity Games
2. Playing (Approximatively) Optimally in LTL Games
3. More Tradeoffs
4. Conclusion
Tradeoffs in infinite games exist:

- Optimality can be prohibitively expensive, both in terms of computational complexity and in terms of memory requirements.
Conclusion

Tradeoffs in infinite games exist:

- Optimality can be prohibitively expensive, both in terms of computational complexity and in terms of memory requirements.

Positive results:

- Optimal bounds in PROMPT–LTL games can be approximated at no extra cost.
- The expressiveness of LTL can be increased considerably for free.
- Lookahead allows to improve strategies and decrease memory requirements.
Conclusion

Tradeoffs in infinite games exist:

- Optimality can be prohibitively expensive, both in terms of computational complexity and in terms of memory requirements.

Positive results:

- Optimal bounds in PROMPT–LTL games can be approximated at no extra cost.
- The expressiveness of LTL can be increased considerably for free.
- Lookahead allows to improve strategies and decrease memory requirements.

⇒ Need to take tradeoffs into account when solving games.