Tradeoffs in Infinite Games

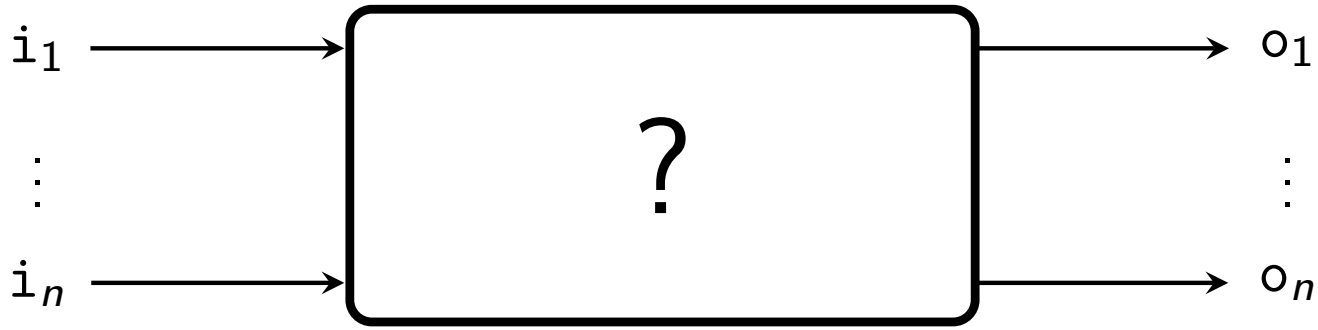
Martin Zimmermann

Saarland University

May 15th, 2018

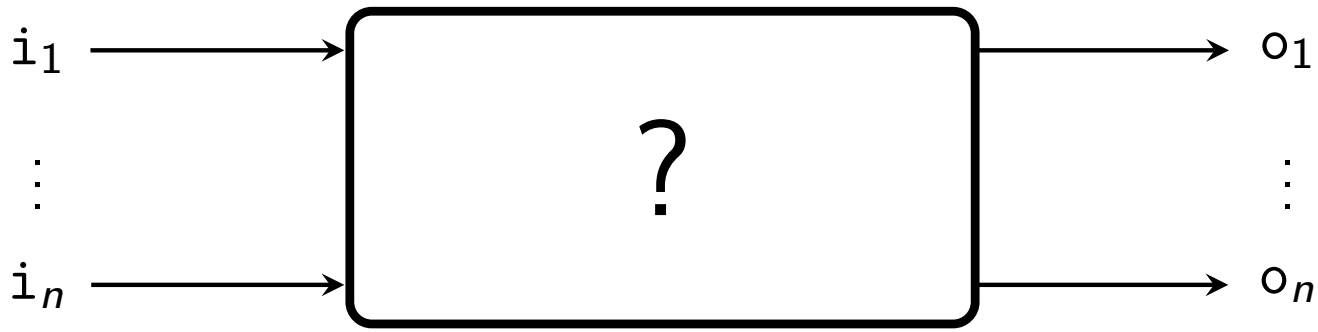
Scientific Talk
in the Habilitation Process

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



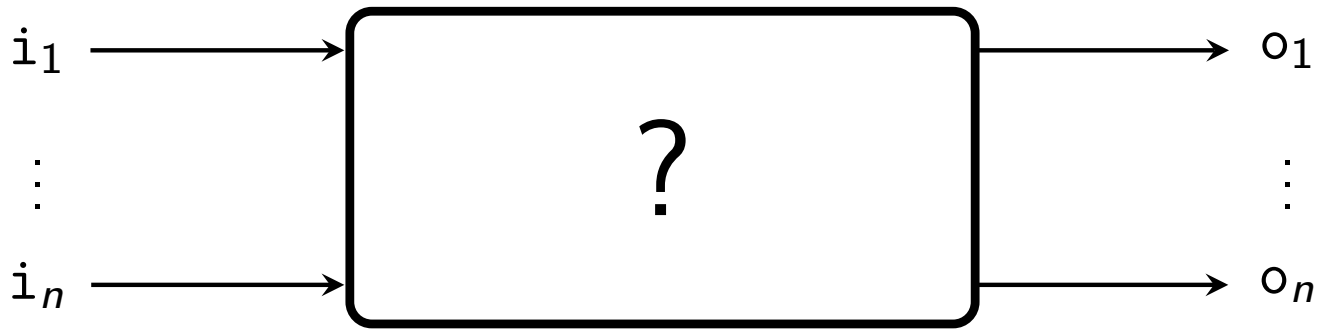
Example

Interpret input $\mathtt{i}_j = 1$ as client j requesting a shared resource and output $\mathtt{o}_j = 1$ 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.

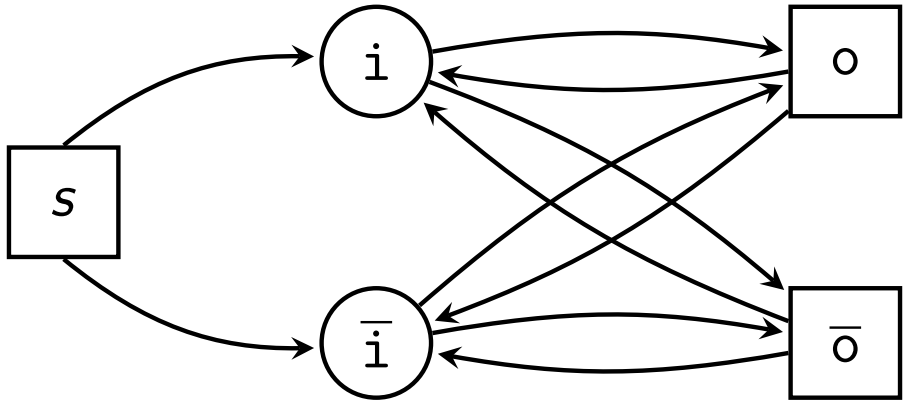
Insight: Problem can be expressed as two-player game of infinite duration between the environment (producing inputs) and the circuit (producing outputs).

Back to the Example

Consider the one-client case!

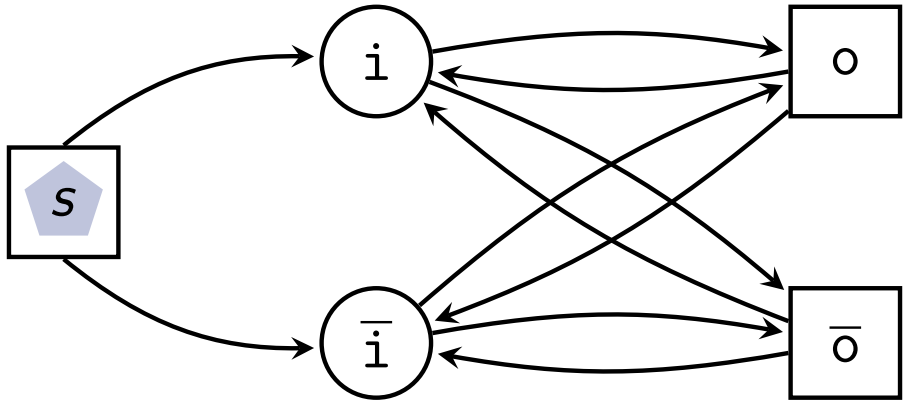
Back to the Example

Consider the one-client case!



Back to the Example

Consider the one-client case!

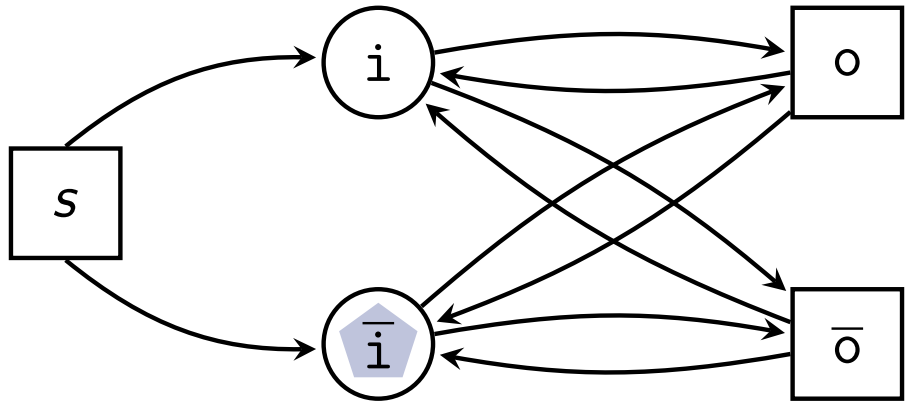


Input:

Output:

Back to the Example

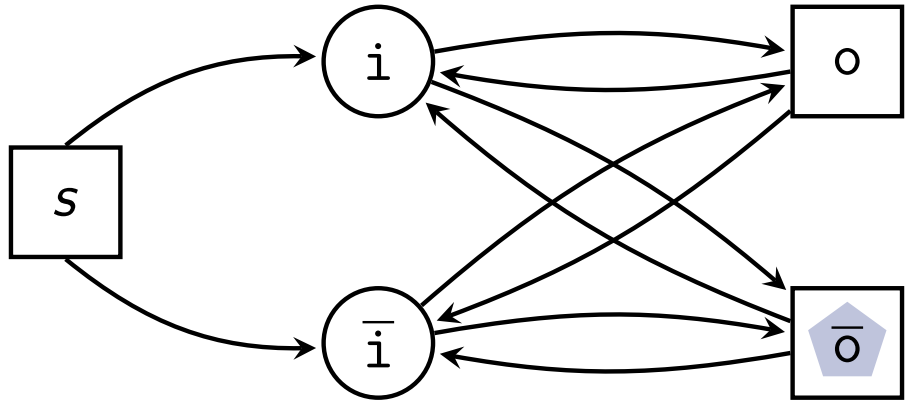
Consider the one-client case!



Input: 0
Output:

Back to the Example

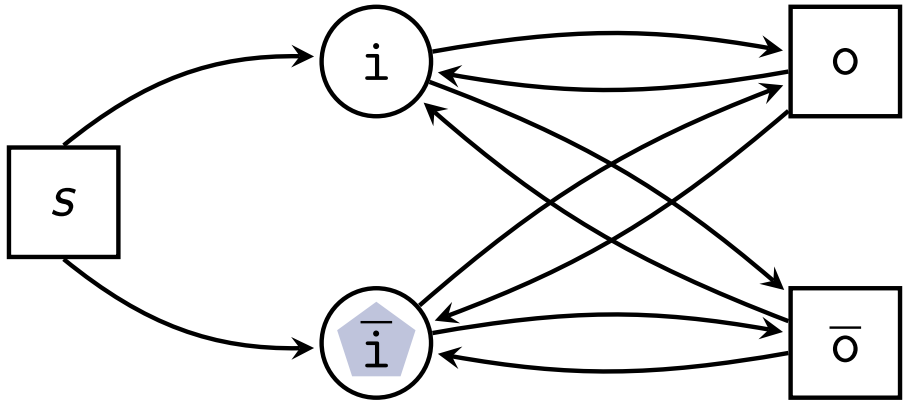
Consider the one-client case!



Input: 0
Output: 0

Back to the Example

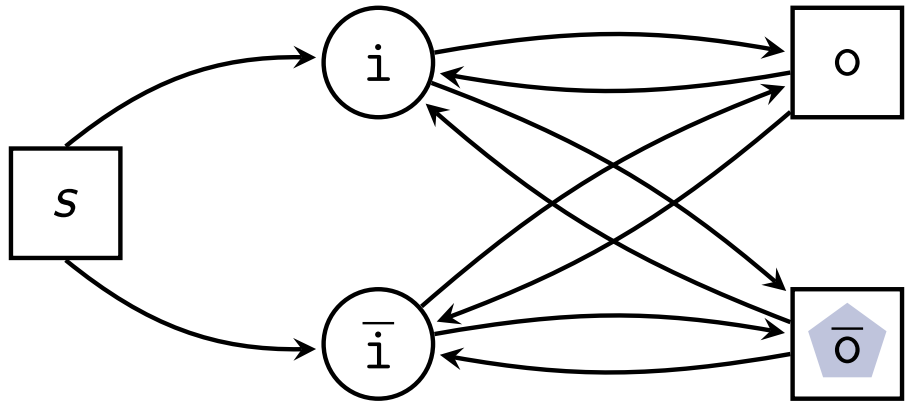
Consider the one-client case!



Input: 0 0
Output: 0

Back to the Example

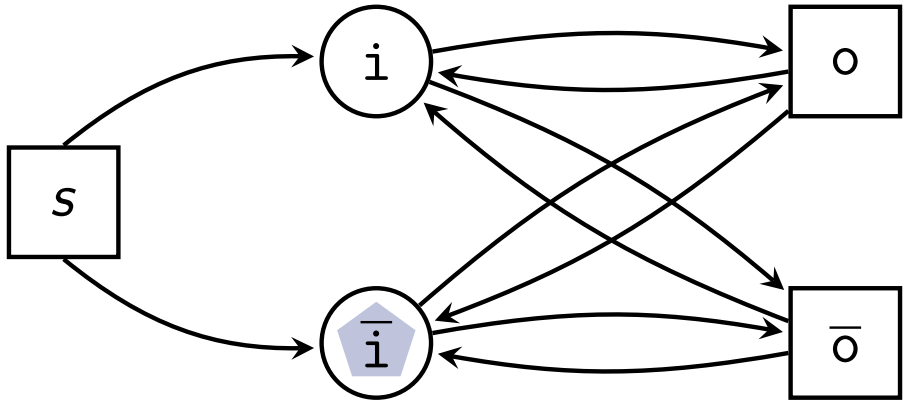
Consider the one-client case!



Input: 0 0
Output: 0 0

Back to the Example

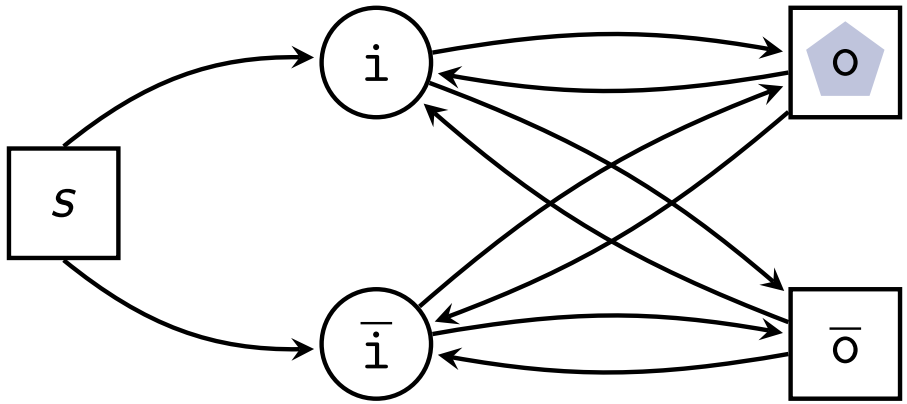
Consider the one-client case!



Input: 0 0 0
Output: 0 0

Back to the Example

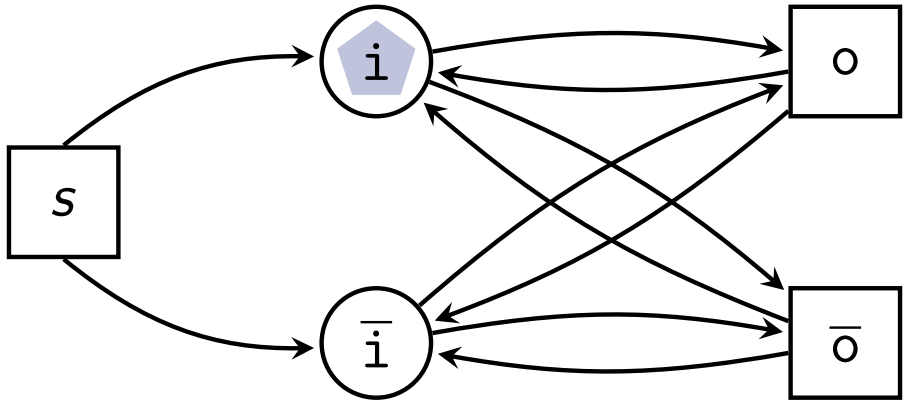
Consider the one-client case!



Input: 0 0 0
Output: 0 0 1

Back to the Example

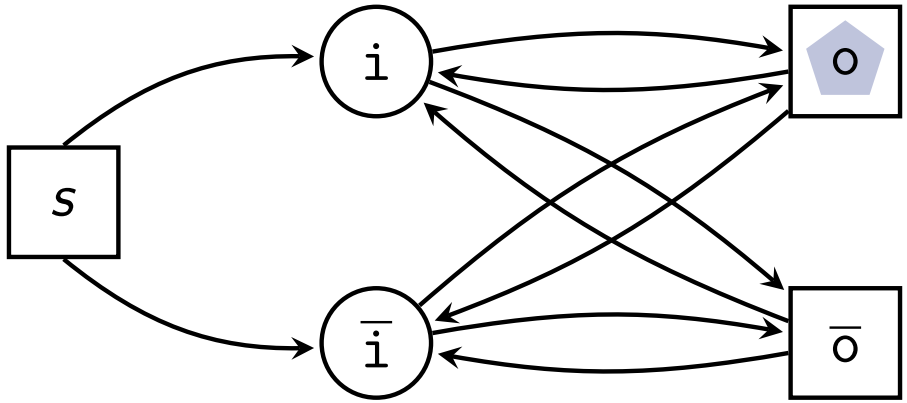
Consider the one-client case!



Input: 0 0 0 1
Output: 0 0 1

Back to the Example

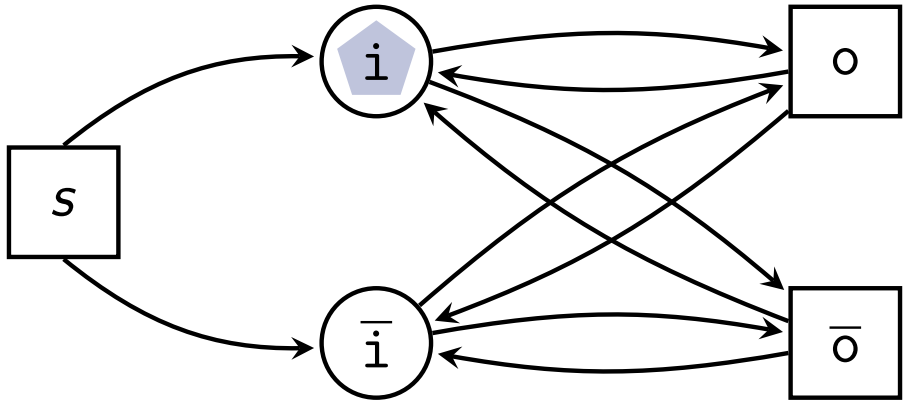
Consider the one-client case!



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

Back to the Example

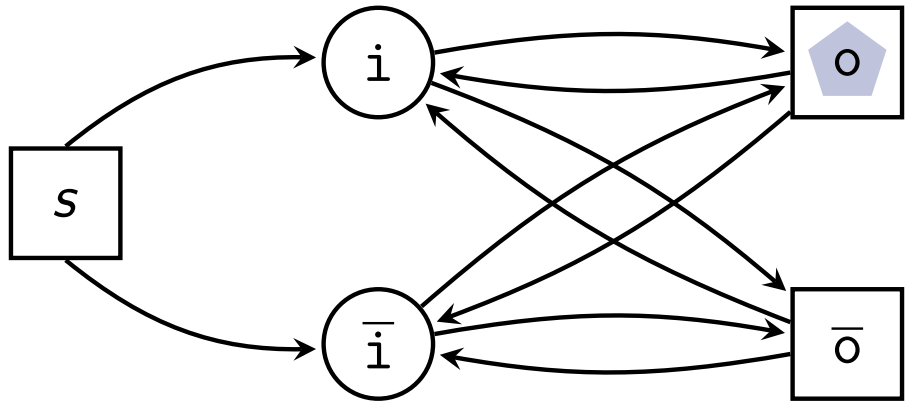
Consider the one-client case!



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

Back to the Example

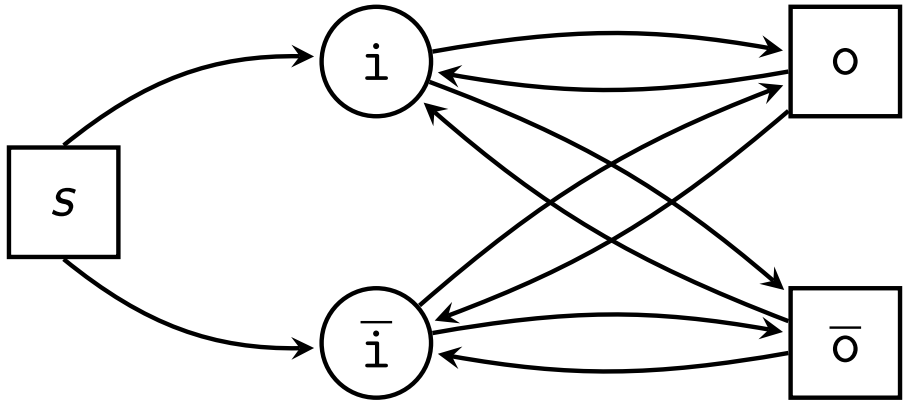
Consider the one-client case!



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

Back to the Example

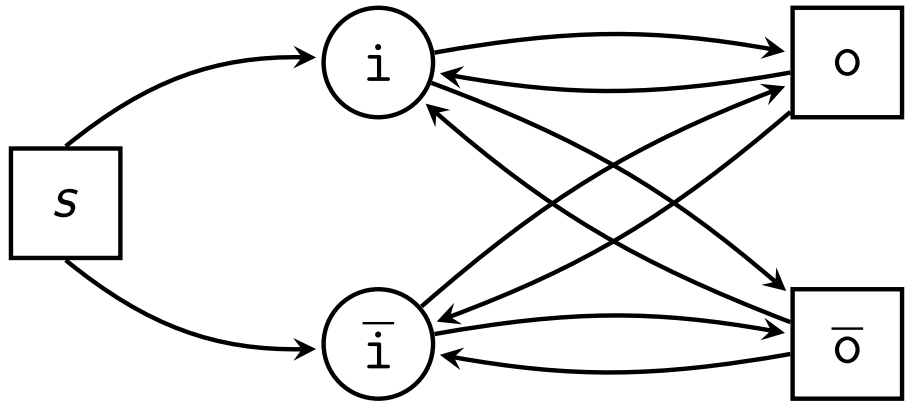
Consider the one-client case!



Input: 0 0 0 1 1 $\cdots$
Output: 0 0 1 1 1 $\cdots$

Back to the Example

Consider the one-client case!



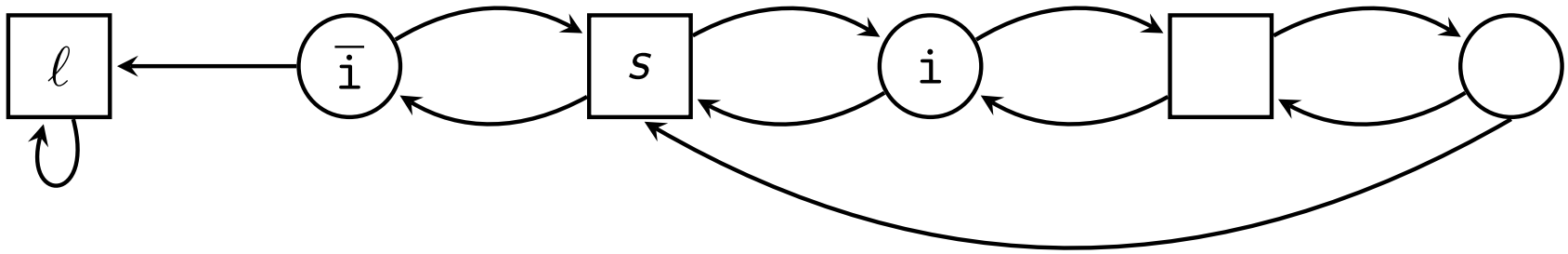
Winning plays for circuit player have to satisfy

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).

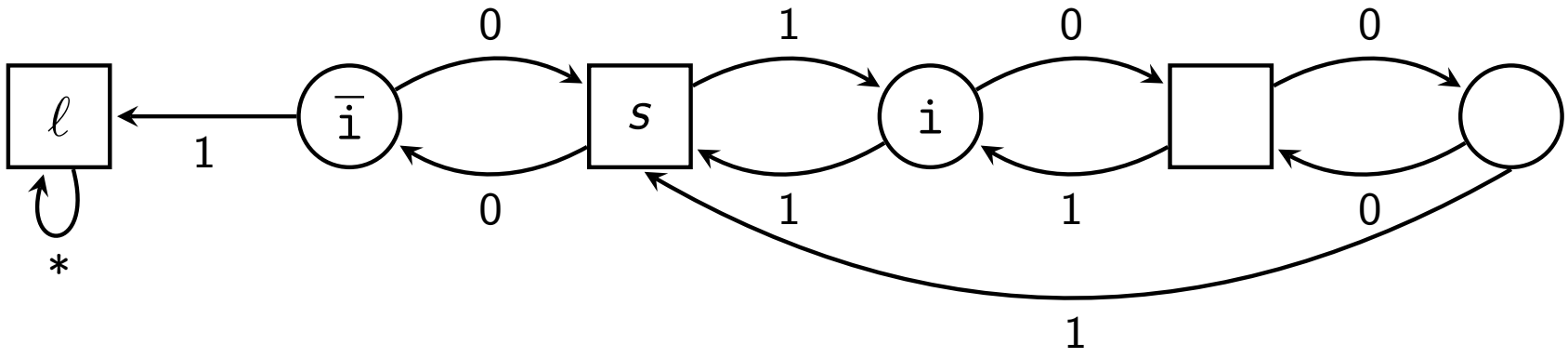
Back to the Example

Consider the one-client case!



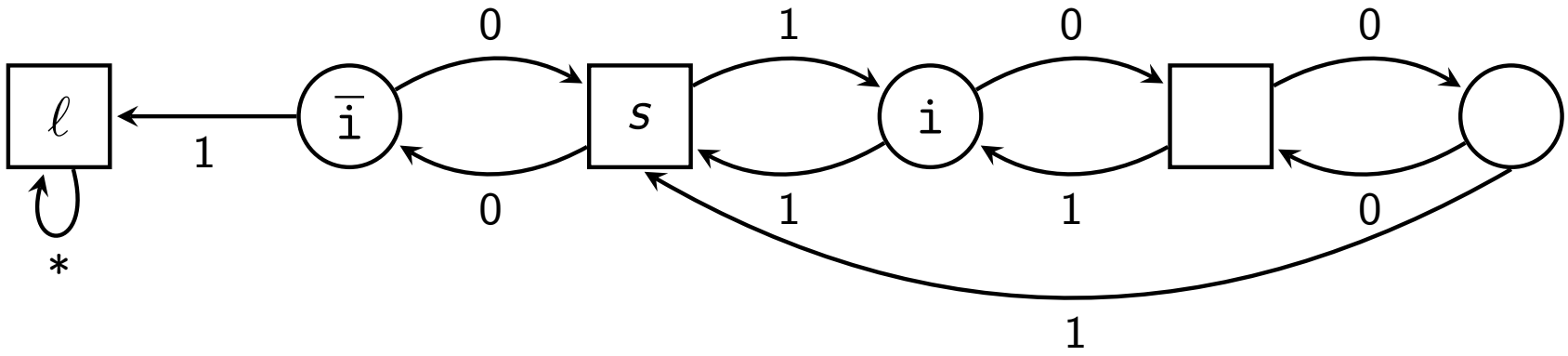
Back to the Example

Consider the one-client case!



Back to the Example

Consider the one-client case!

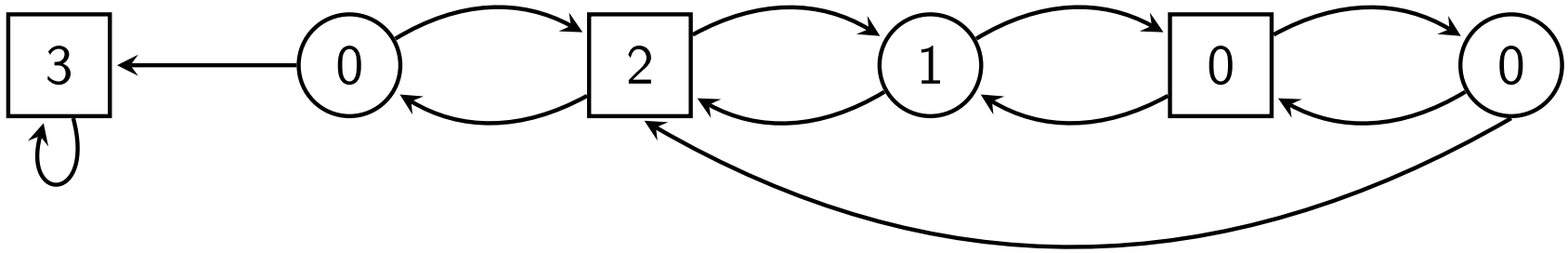


Now, the winning plays for the circuit player have to satisfy

1. if $\mathtt{i}$ is visited infinitely often, then s as well, and
2. ℓ is never visited.

Back to the Example

Consider the one-client case!



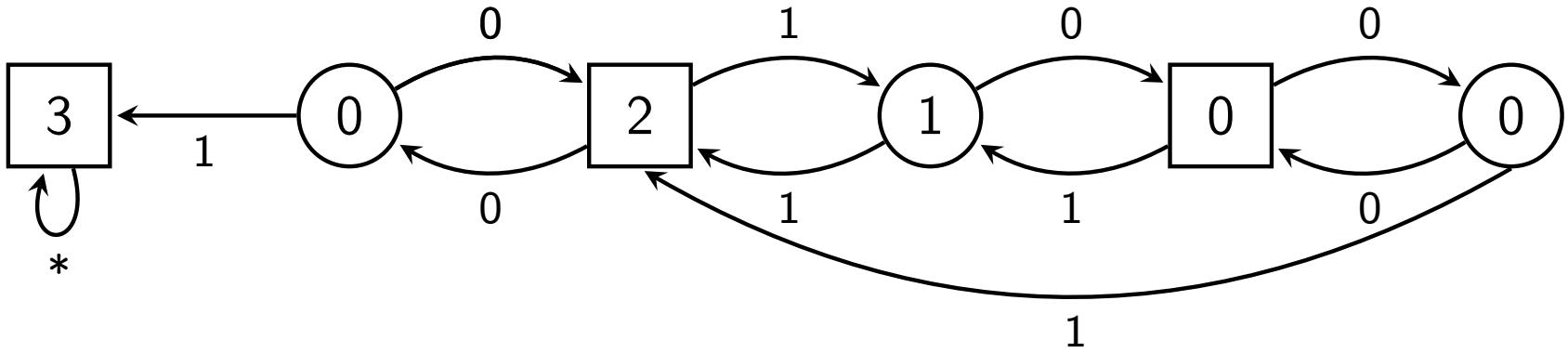
Now, the winning plays for the circuit player have to satisfy

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

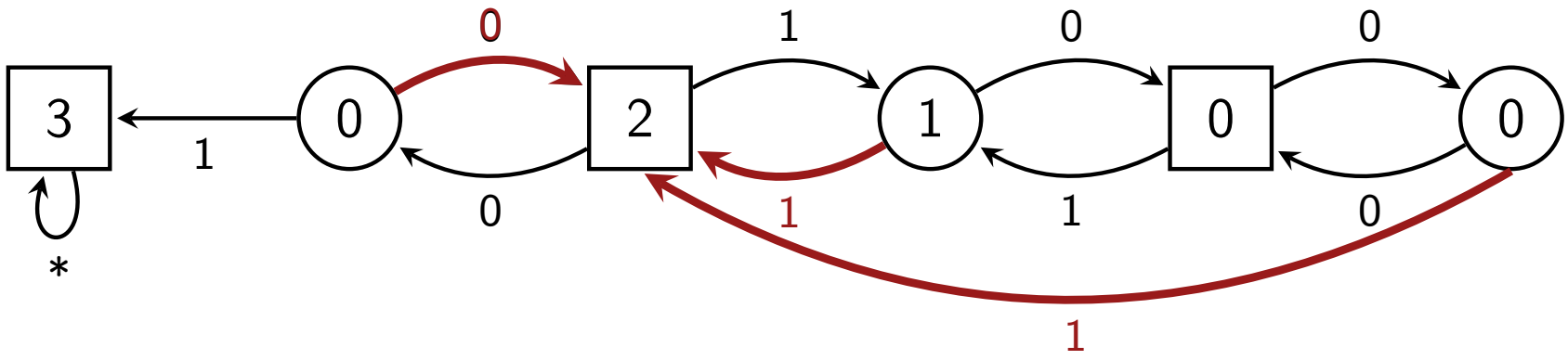
Equivalently: color the vertices by natural numbers as above and require that almost all odd colors are followed by a larger even one.

This is the classical parity condition for ω-automata.

Büchi-Landweber in a Nutshell

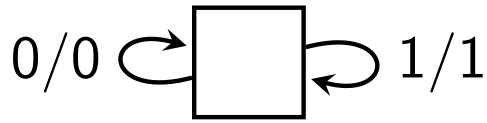


Büchi-Landweber in a Nutshell



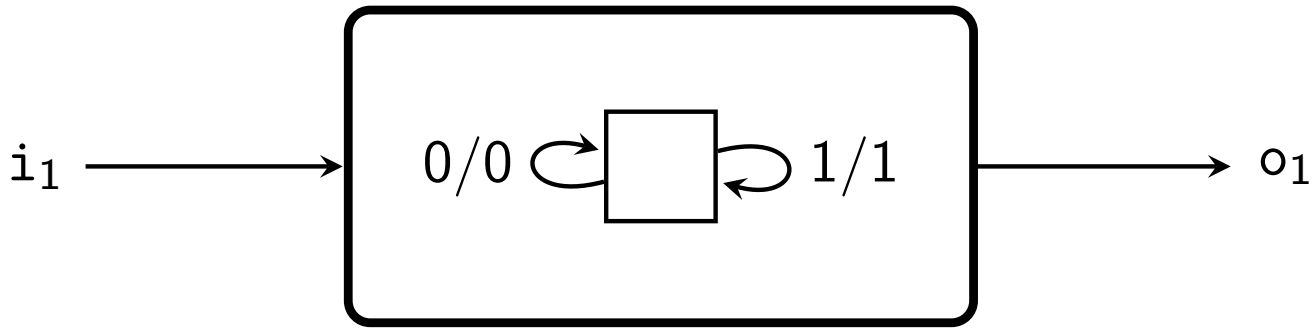
- Player 0 has a memoryless winning strategy,

Büchi-Landweber in a Nutshell



- Player 0 has a memoryless winning strategy,
- which can be turned into an automaton with output,

Büchi-Landweber in a Nutshell



- 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.

Not Everyone is Equal

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.

Not Everyone is Equal

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?

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?

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?

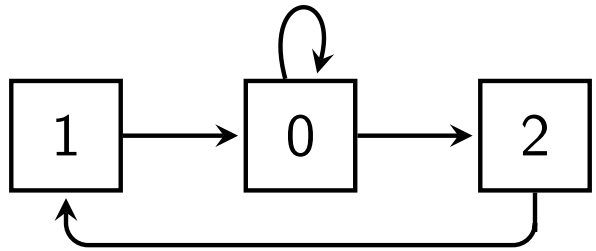
Outline

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**

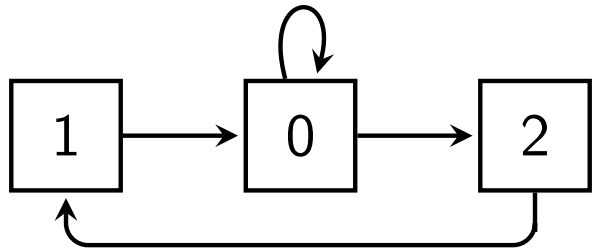
Outline

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

A Parity Game



A Parity Game

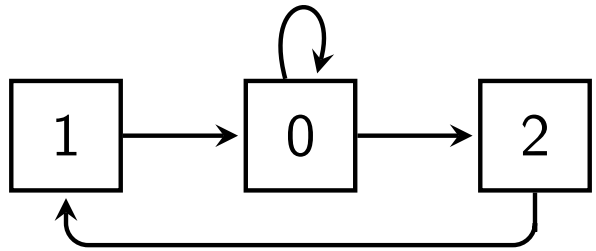


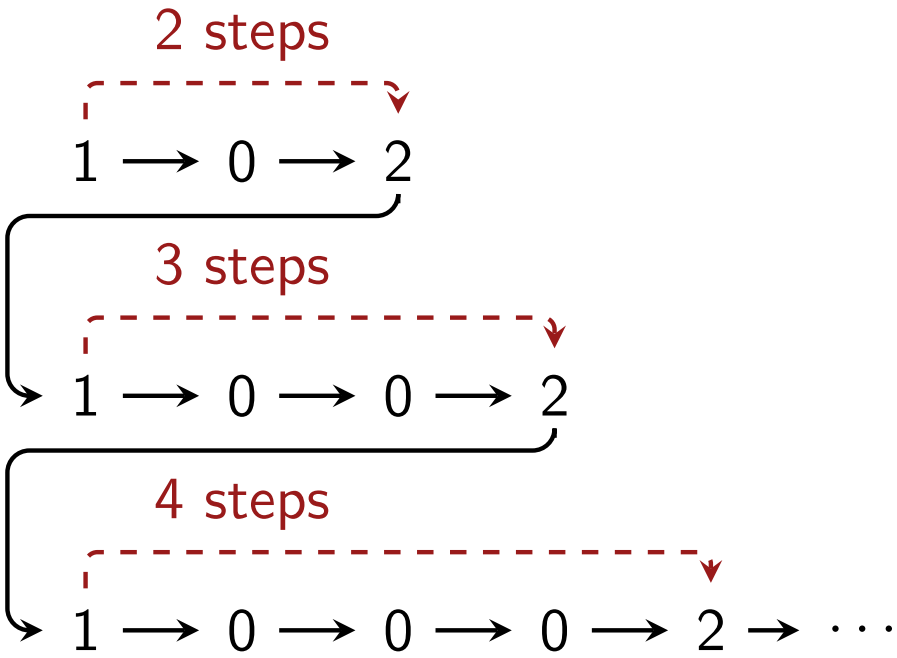
$1 \longrightarrow 0 \longrightarrow 2$

$1 \longrightarrow 0 \longrightarrow 0 \longrightarrow 2$

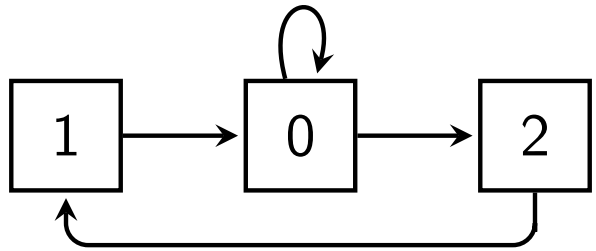
$1 \longrightarrow 0 \longrightarrow 0 \longrightarrow 0 \longrightarrow 2 \rightarrow \cdots$

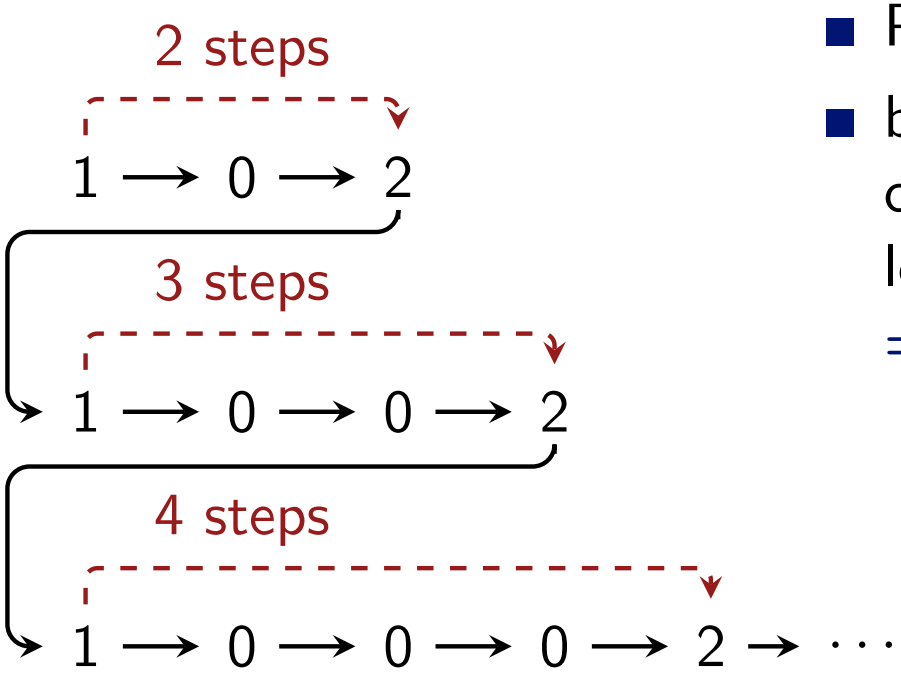
A Parity Game





A Parity Game





- Player 0 wins from every vertex,
- but Player 1 can delay between color 1 and color 2 longer and longer.
  $\Rightarrow$ undesired behavior.

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]**

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]**

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.

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.

Condition	Complexity	Memory Pl. 0	Memory Pl. 1
Parity	quasi-poly	Memoryless	Memoryless
Finitary Parity	PTime	Memoryless	Infinite

Boundedness vs. Optimization

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.

Boundedness vs. Optimization

The bound b in the definition of finitary parity games is existentially quantified (and may depend on the play).

Corollary

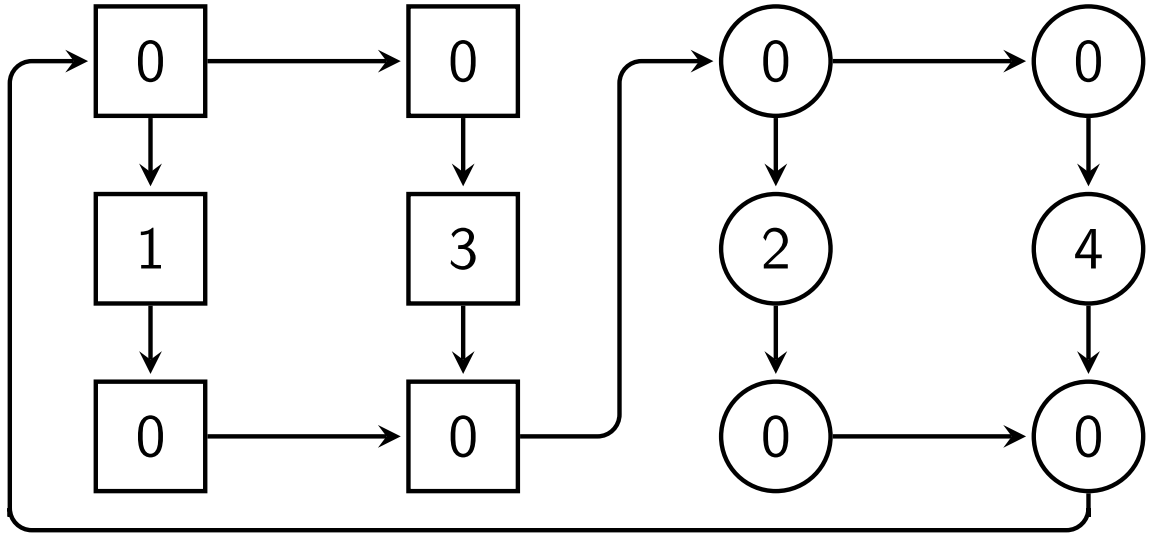
If Player 0 *wins a finitary parity game* $\mathcal{G}$*, then a uniform bound* $b \leq |\mathcal{G}|$ *suffices.*

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

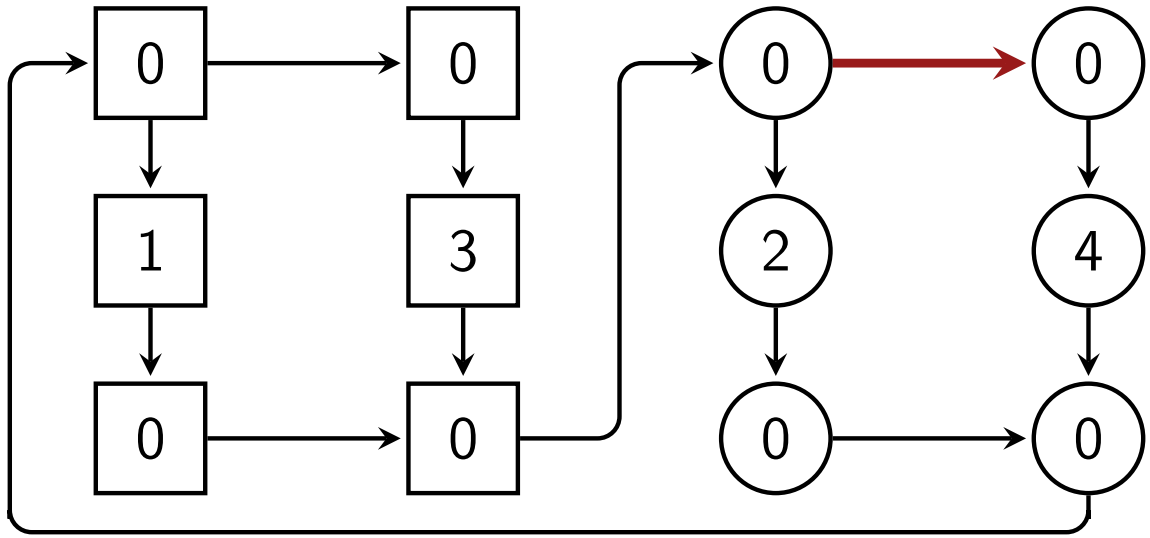
Questions

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?

Chatterjee & Fijalkow

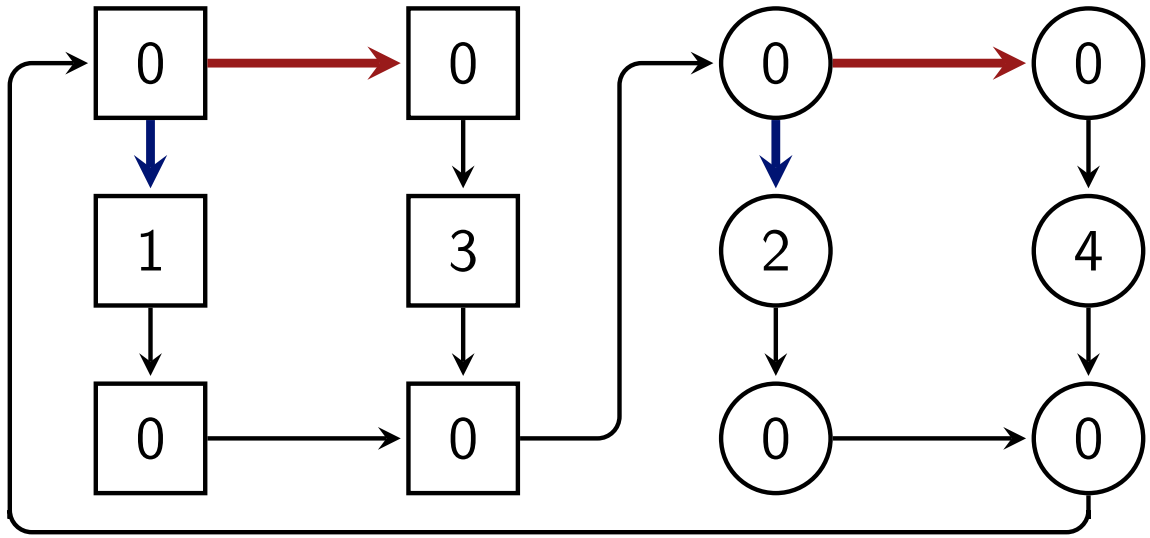


Chatterjee & Fijalkow



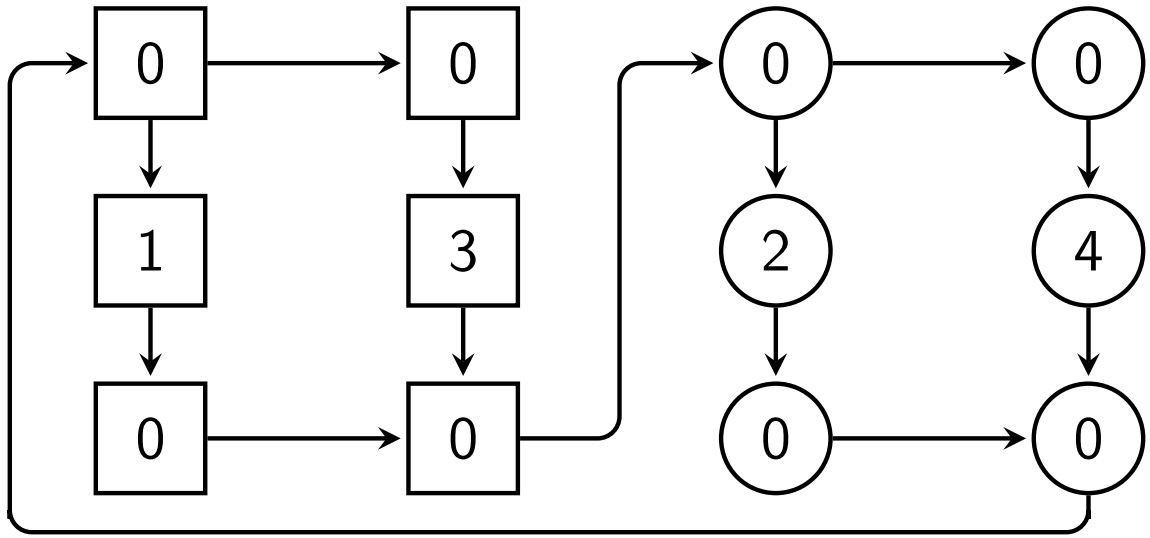
- Player 0 has a unique memoryless winning strategy, which achieves the bound five: from the 1 it takes five steps to the 4.

Chatterjee & Fijalkow



- 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.

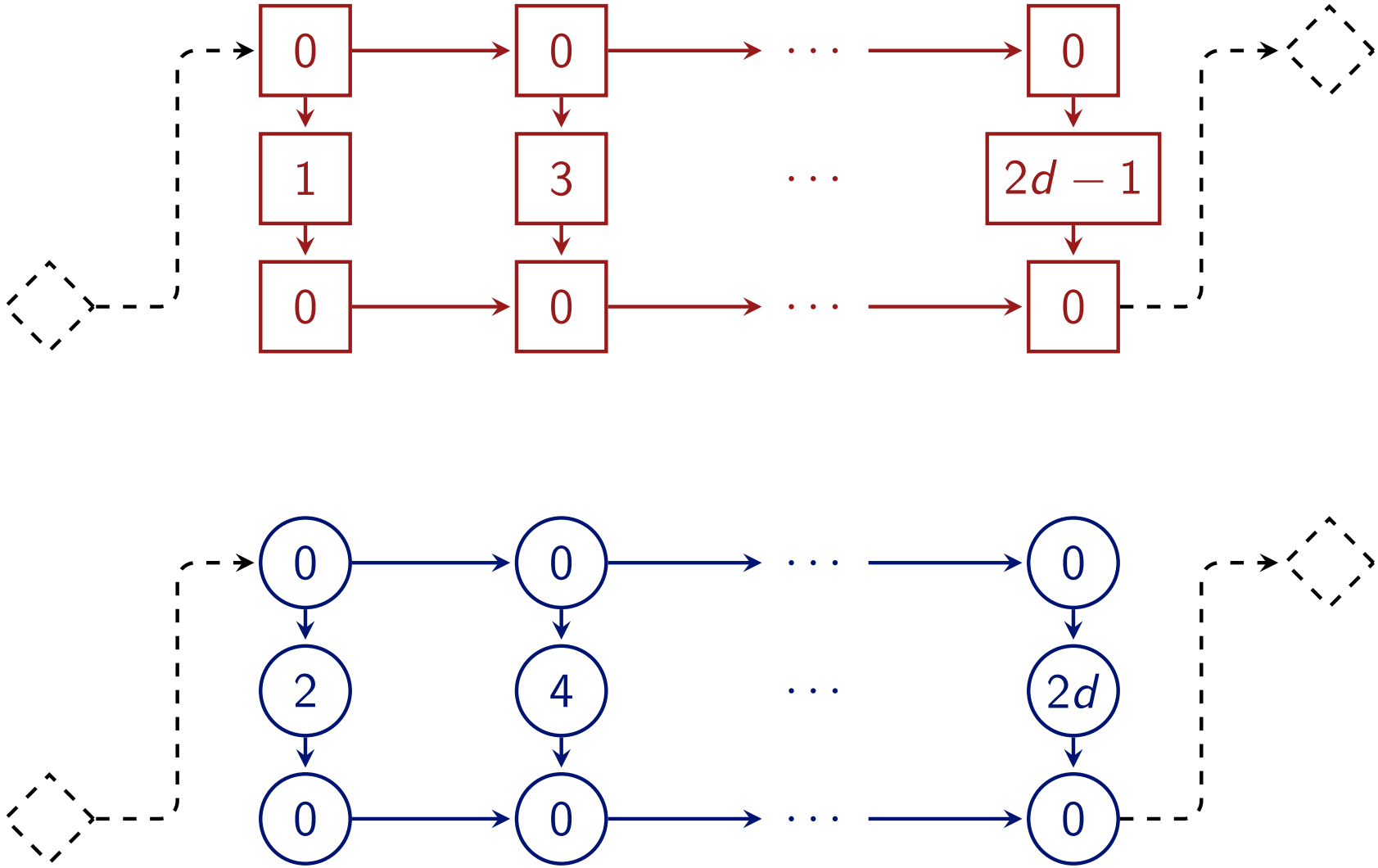
Chatterjee & Fijalkow



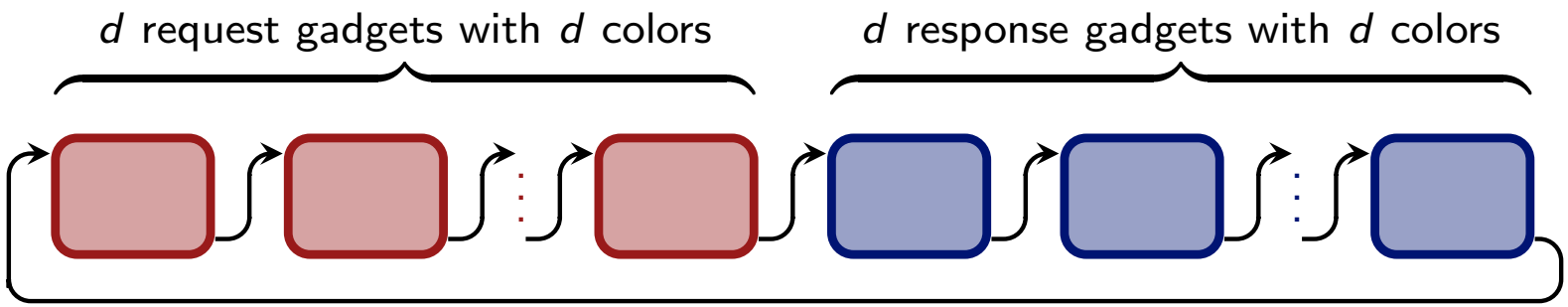
- Player 0 has a unique memoryless winning strategy, which achieves the bound five: from the 1 it takes five steps to the 4.
- With two memory states, she can achieve the bound four: "answer" a 1 by a 2 and a 3 by a 4.
- It is trivial to extend this example to d odd colors and d even colors requiring d memory states to play optimally.

$\Rightarrow$ In general, playing optimally requires memory; but how much?

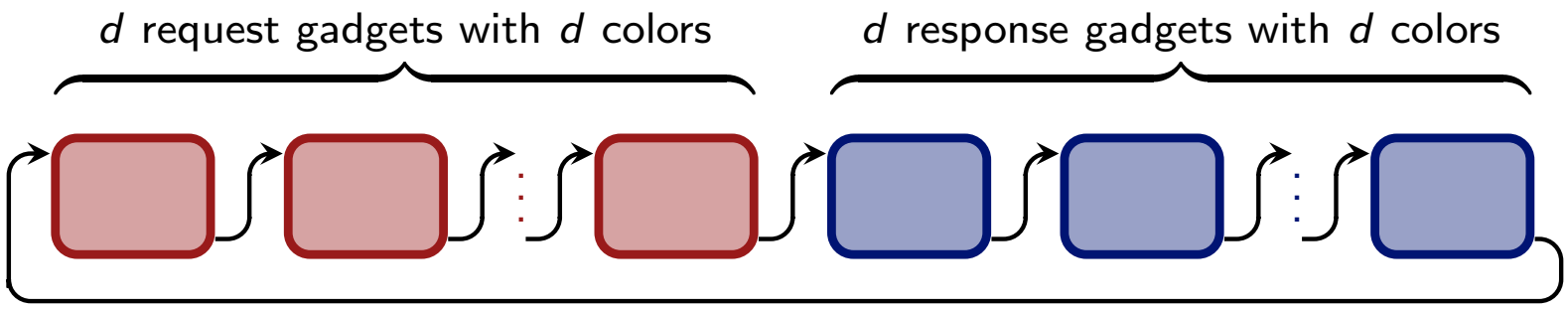
Memory Requirements



Memory Requirements

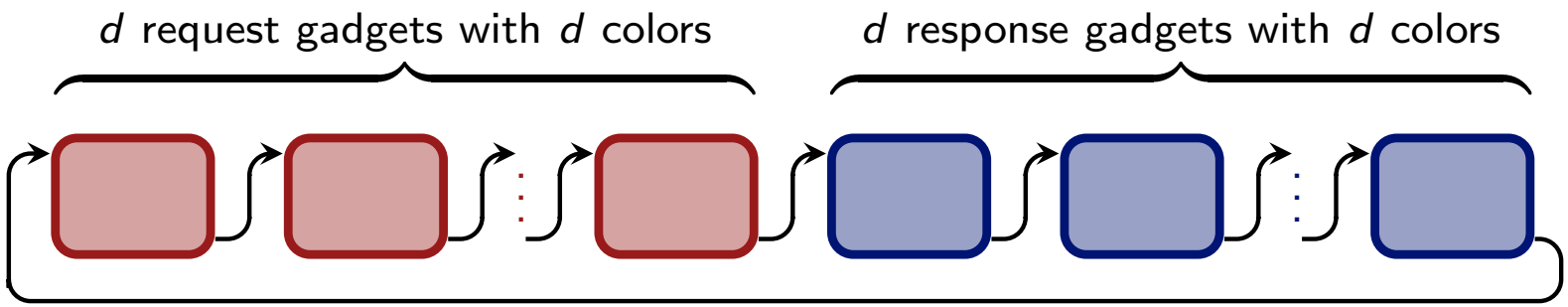


Memory Requirements



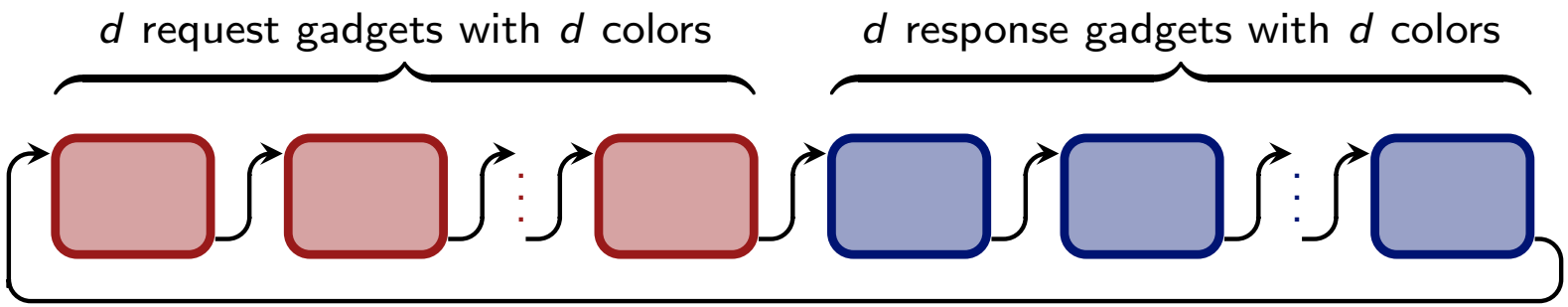
- Player 0 has winning strategy with cost $d^2 + 2d$: answer j-th unique request in j-th response-gadget.
  $\Rightarrow$ requires exponential memory (in d).

Memory Requirements



- 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.
  $\Rightarrow$ 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 $\mathcal{G}_d$ such that

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

PSPACE-Hardness

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*?"*

PSPACE-Hardness

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*?"*

Proof

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

PSPACE-Hardness

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*?"*

Proof

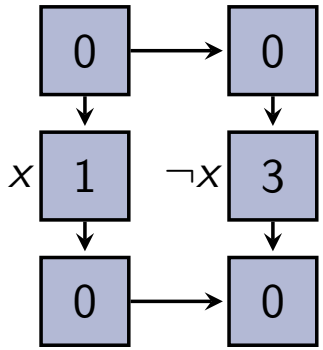
- By a reduction from QBF (w.l.o.g. in CNF).
- Checking the truth of $\varphi = \forall x \exists y.\ (x \vee \neg y) \wedge (\neg x \vee y)$ as a two-player game (Player 0 wants to prove truth of φ):
  1. Player 1 picks truth value for x.
  2. Player 0 picks truth value for y.
  3. Player 1 picks clause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

The Reduction

$$\varphi = \forall x \ \exists y \,.\ \overbrace{(\, x \ \vee \ \neg y \,) \wedge (\, \neg x \ \vee \ y \,)}^{\psi}$$

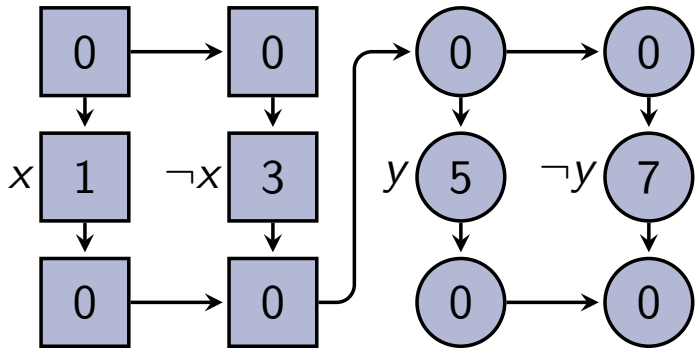
The Reduction

$$\varphi = \forall x \; \exists y \; . \; \overbrace{(x \vee \neg y) \wedge (\neg x \vee y)}^{\psi}$$



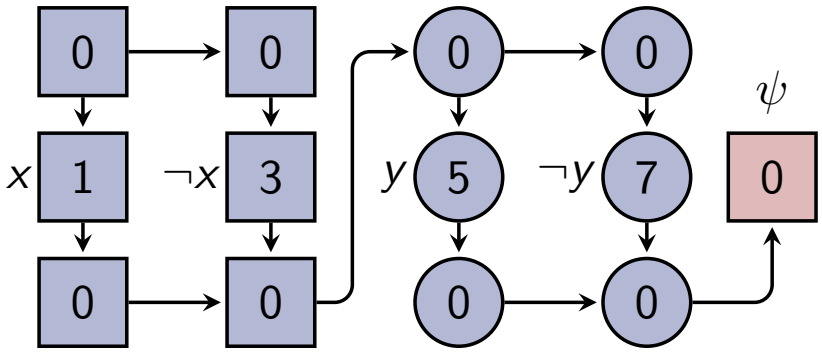
The Reduction

$$\varphi = \forall x \, \exists y \, . \, \overbrace{(x \vee \neg y) \wedge (\neg x \vee y)}^{\psi}$$



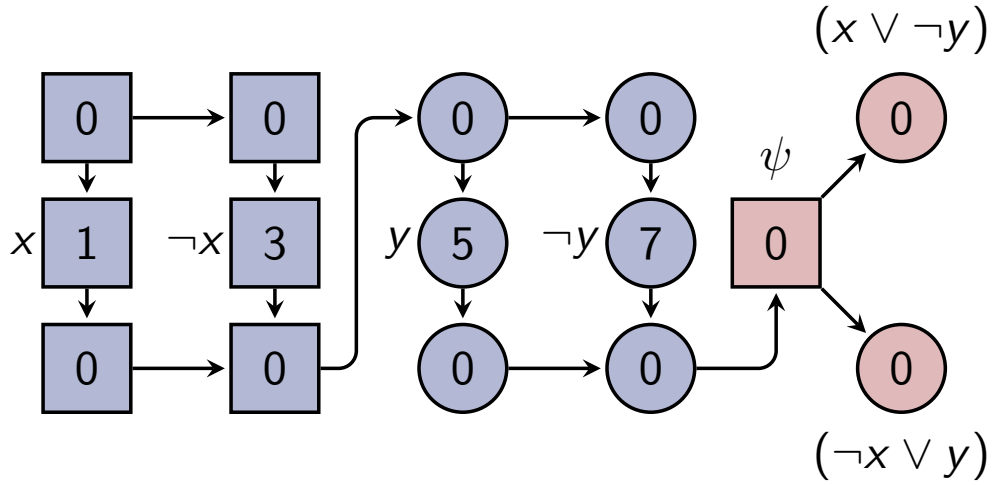
The Reduction

$$\varphi = \forall x \ \exists y \ . \ \overbrace{(x \ \vee \ \neg y \ ) \wedge (\neg x \ \vee \ y \ )}^{\psi}$$



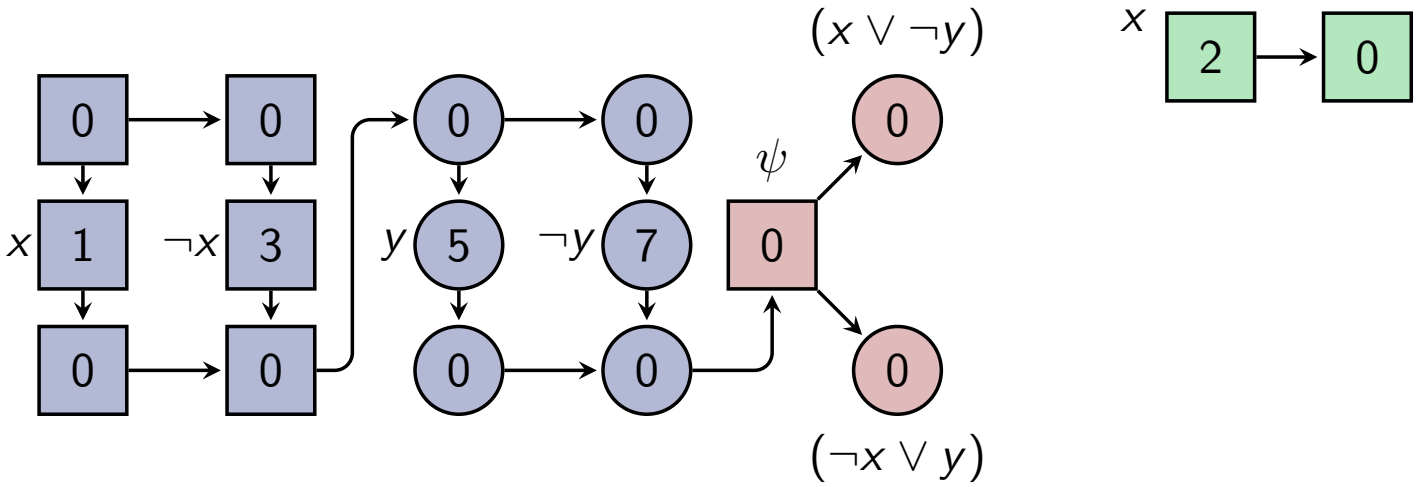
The Reduction

$$\varphi = \forall x \ \exists y \ . \ \overbrace{(x \vee \neg y) \wedge (\neg x \vee y)}^{\psi}$$



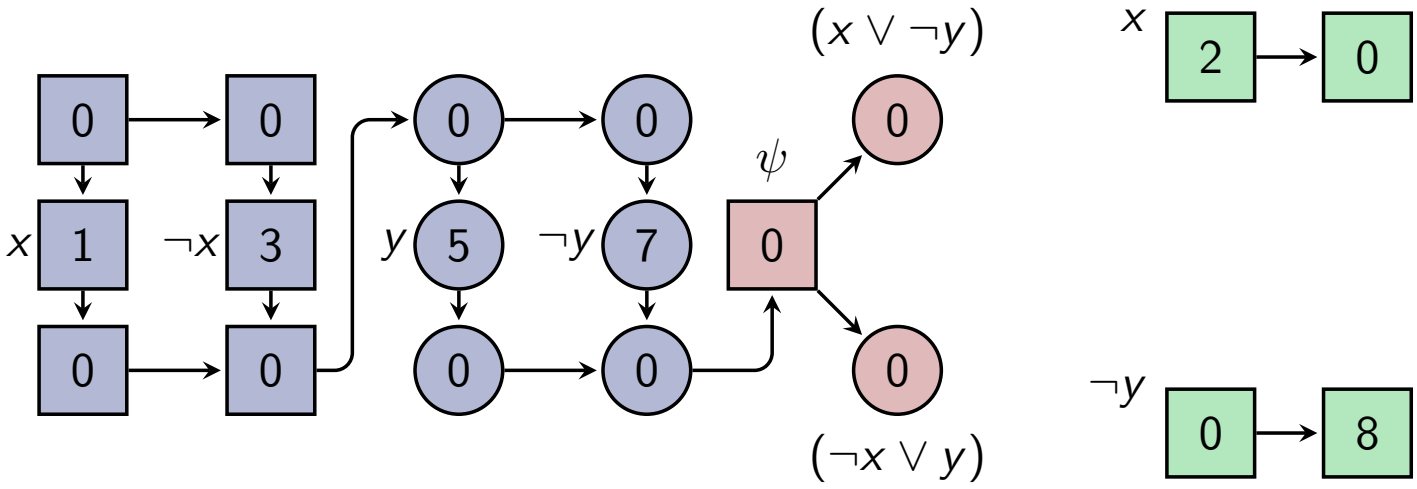
The Reduction

$$\varphi = \forall x \ \exists y \ . \ \overbrace{(\ x \ \vee \ \neg y\ ) \wedge (\ \neg x \ \vee \ y\ )}^{\psi}$$



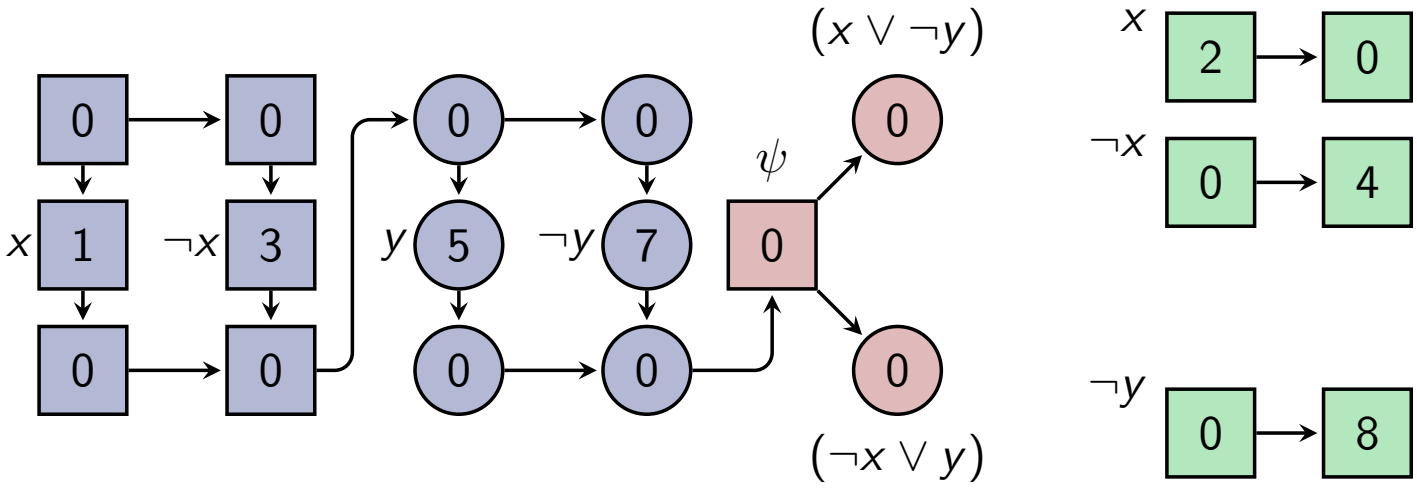
The Reduction

$$\varphi = \forall x \ \exists y \ . \ \overbrace{(x \ \vee \ \neg y \ ) \wedge (\neg x \ \vee \ y \ )}^{\psi}$$



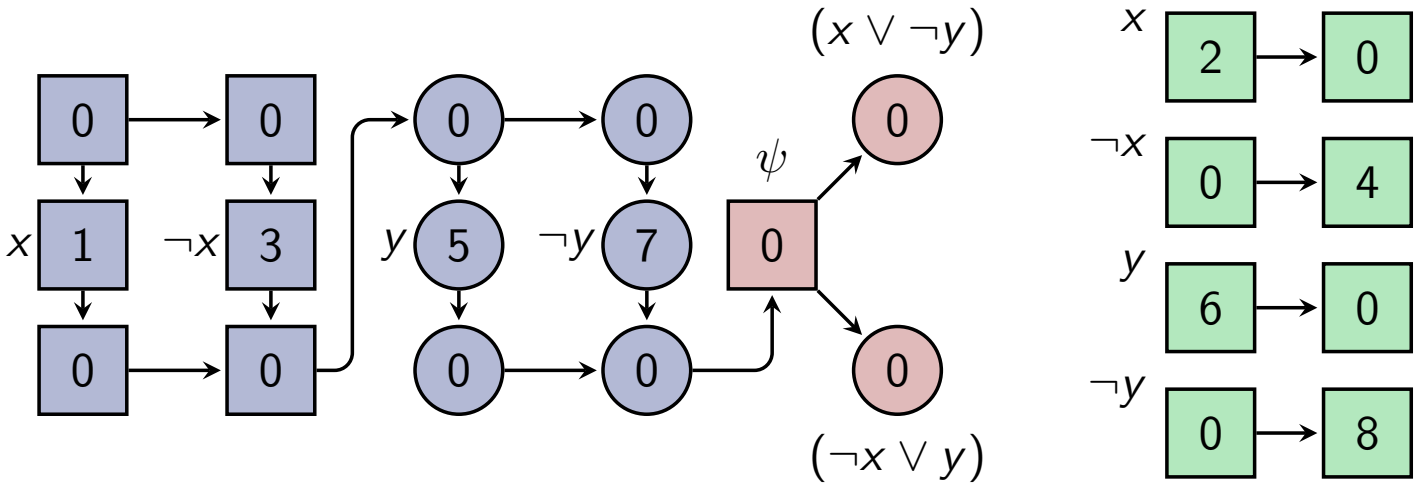
The Reduction

$$\varphi = \forall x \ \exists y \ . \ \overbrace{(x \ \vee \ \neg y \ ) \wedge (\ \neg x \ \vee \ y \ )}^{\psi}$$



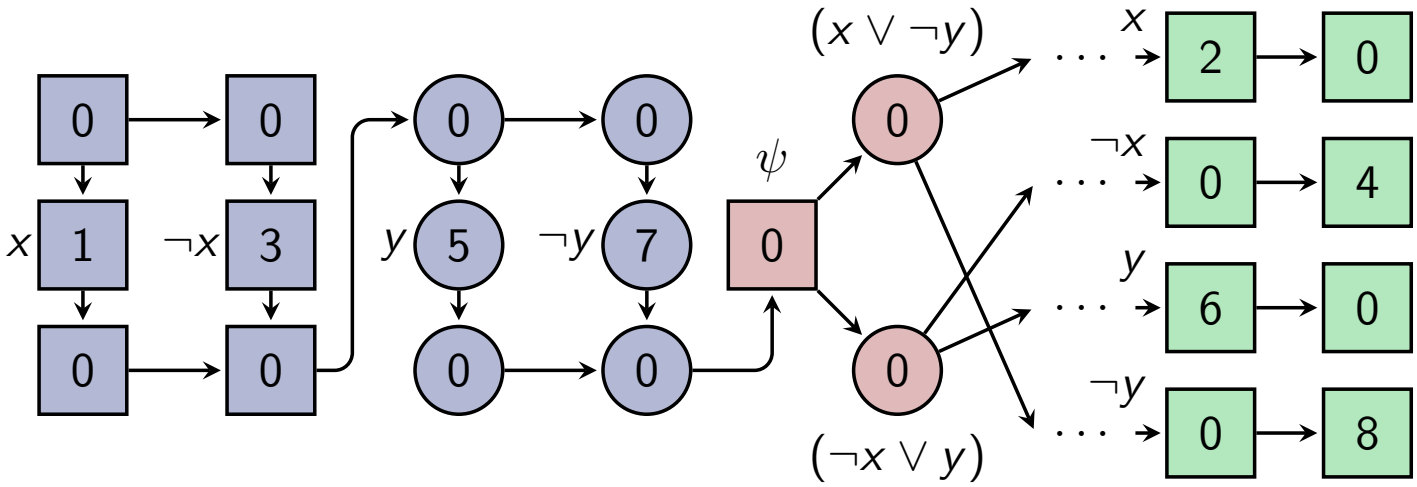
The Reduction

$$\varphi = \forall x \ \exists y \,.\, \overbrace{(\, x \ \vee \ \neg y \,) \wedge (\, \neg x \ \vee \ y \,)}^{\psi}$$



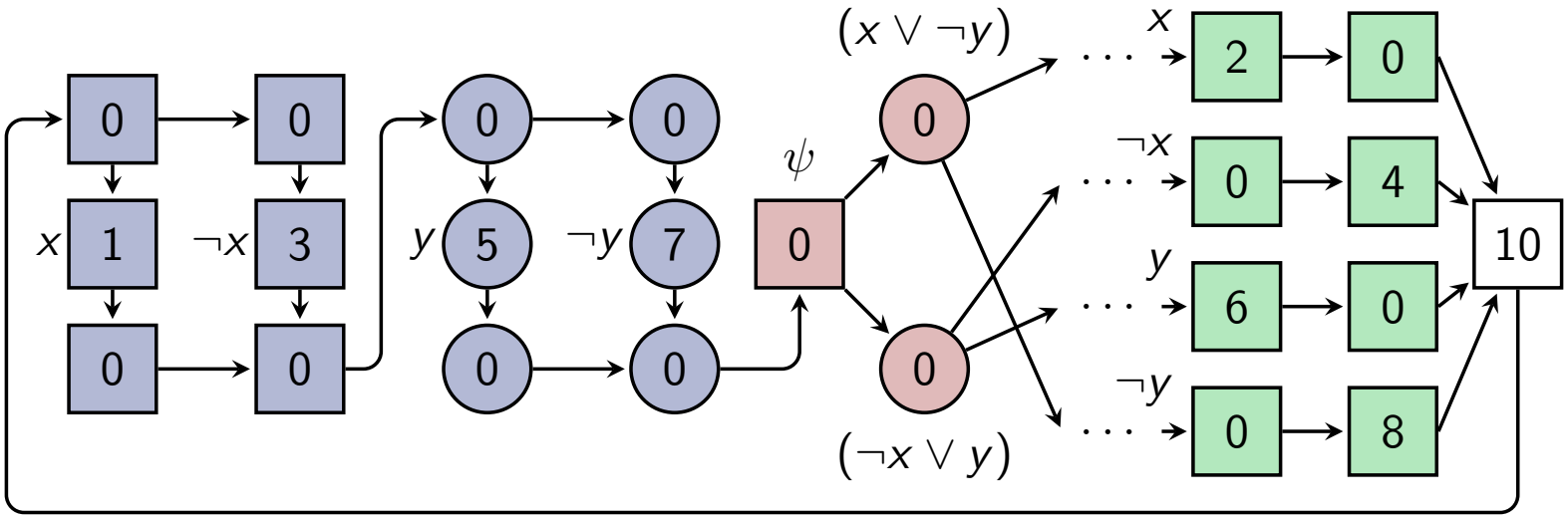
The Reduction

$$\varphi = \forall x \ \exists y \ . \ \overbrace{(x \ \vee \ \neg y \ ) \wedge (\neg x \ \vee \ y \ )}^{\psi}$$



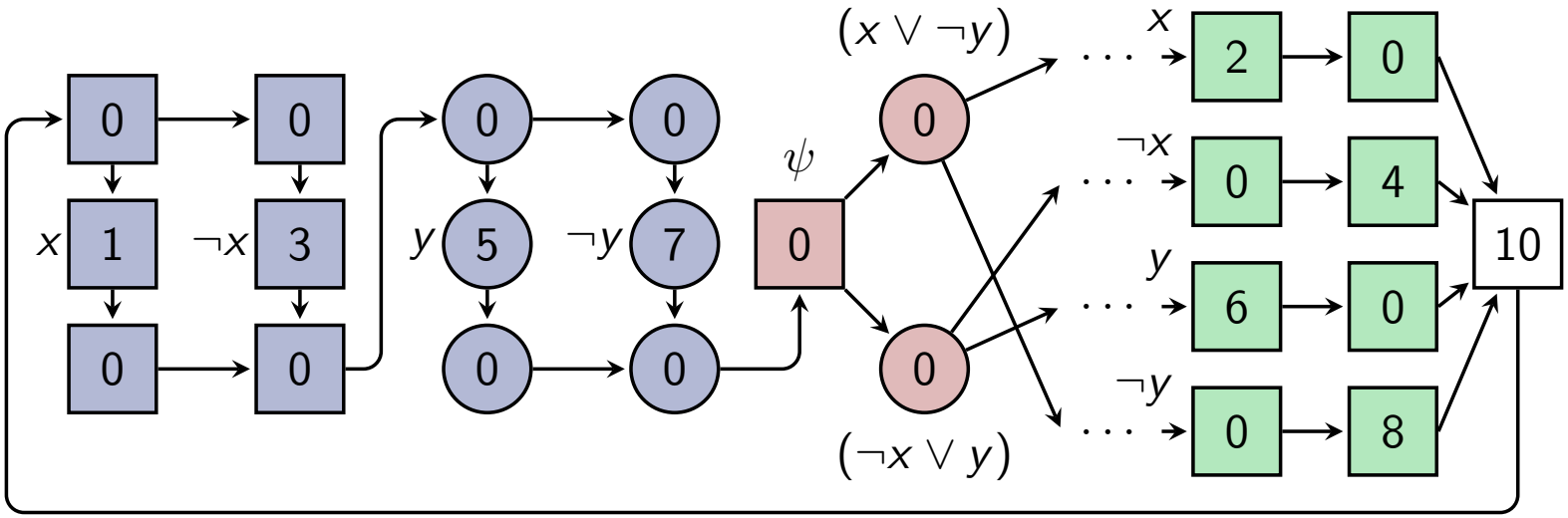
The Reduction

$$\varphi = \forall x \ \exists y \,.\ \overbrace{(\, x \ \vee \ \neg y \,) \wedge (\, \neg x \ \vee \ y \,)}^{\psi}$$



The Reduction

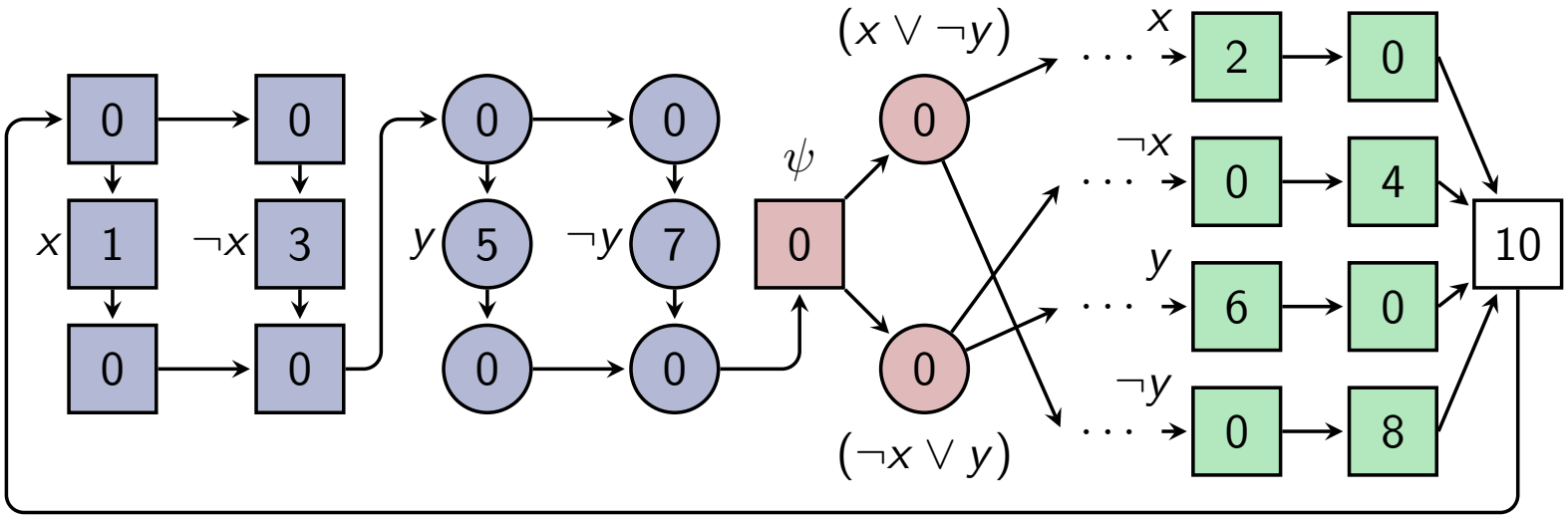
$$\varphi = \forall x \ \exists y \ . \ \overbrace{(x \ \vee \ \neg y \ ) \wedge (\neg x \ \vee \ y \ )}^{\psi}$$



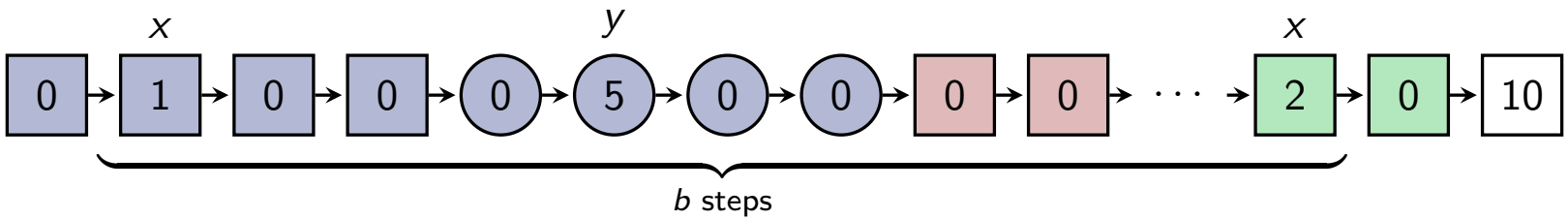
For a well-chosen bound b, a strategy for Player 0 with cost at most b witnesses the truth of φ and vice versa.

The Reduction

$$\varphi = \forall x \ \exists y \ . \ \overbrace{(x \ \vee \ \neg y \ ) \wedge (\neg x \ \vee \ y \ )}^{\psi}$$

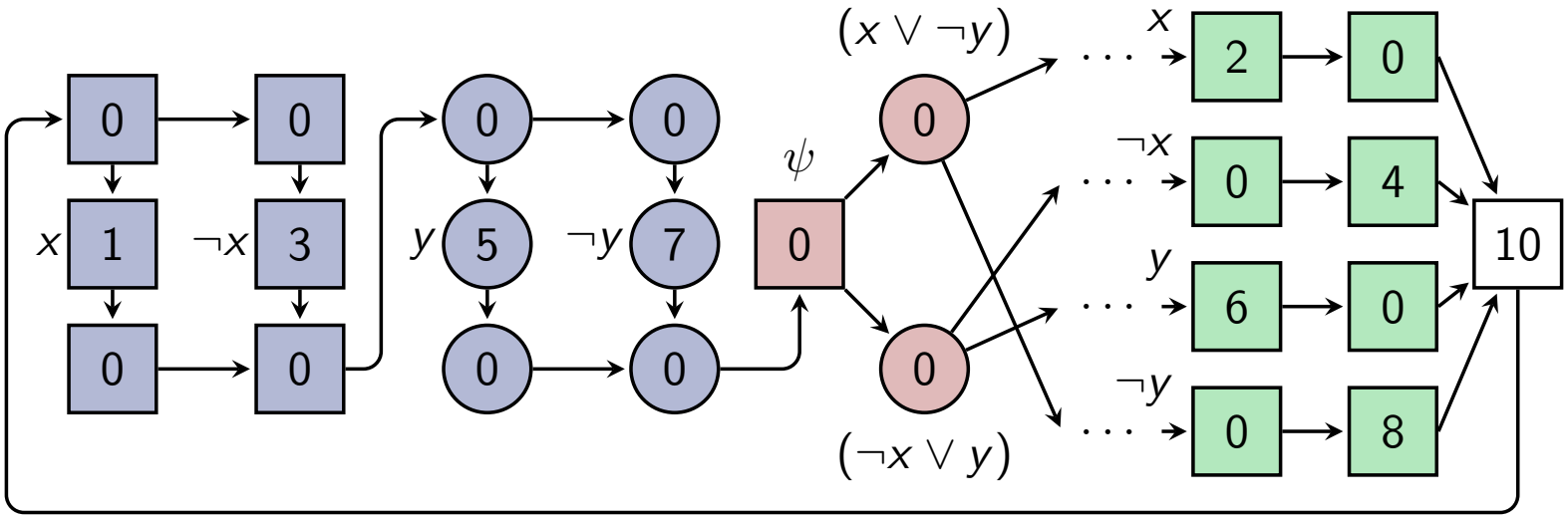


For a well-chosen bound b, a strategy for Player 0 with cost at most b witnesses the truth of φ and vice versa.

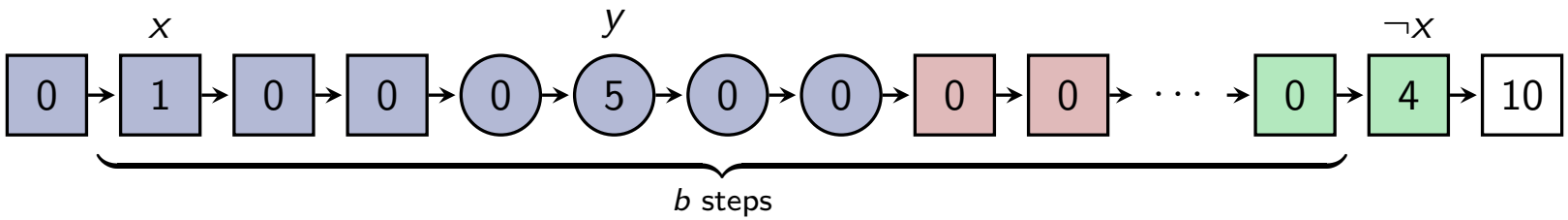


The Reduction

$$\varphi = \forall x \ \exists y \ . \ \overbrace{(x \ \vee \ \neg y \ ) \wedge (\neg x \ \vee \ y \ )}^{\psi}$$

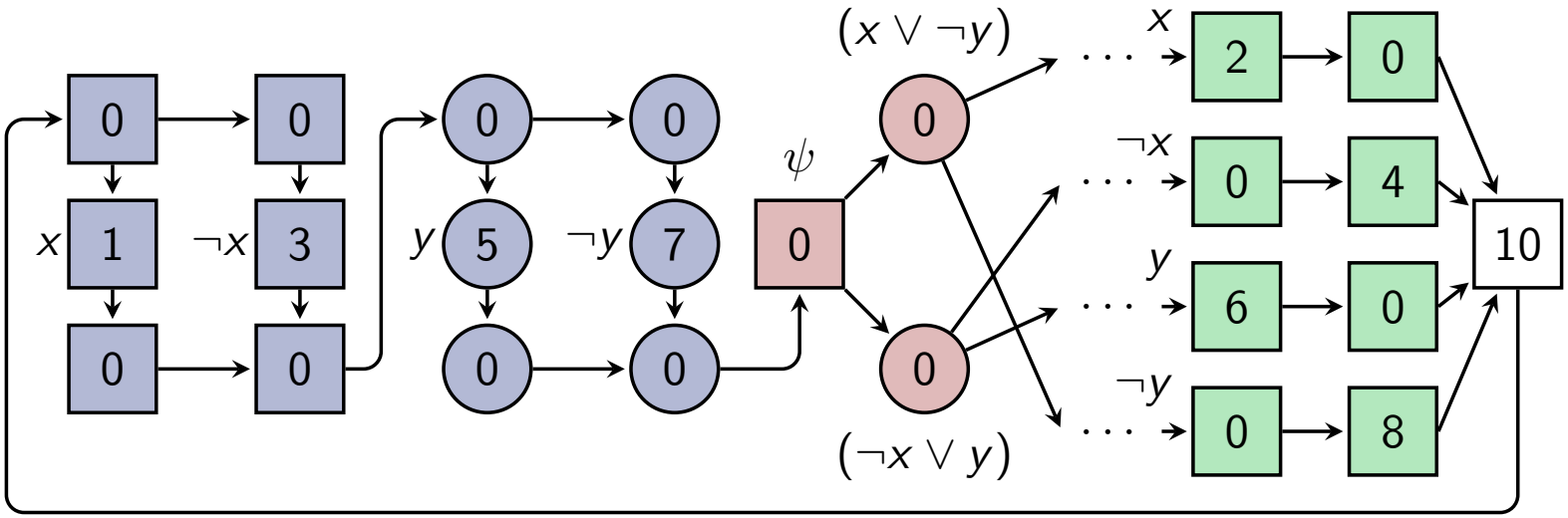


For a well-chosen bound b, a strategy for Player 0 with cost at most b witnesses the truth of φ and vice versa.

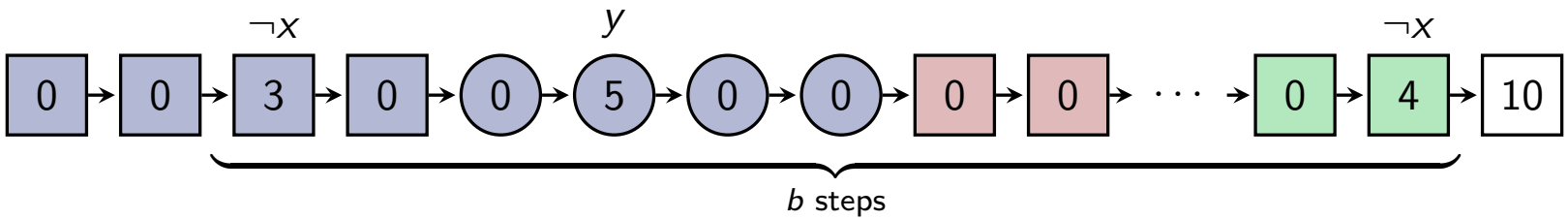


The Reduction

$$\varphi = \forall x \ \exists y \ . \ \overbrace{(x \ \vee \ \neg y \ ) \wedge (\neg x \ \vee \ y \ )}^{\psi}$$

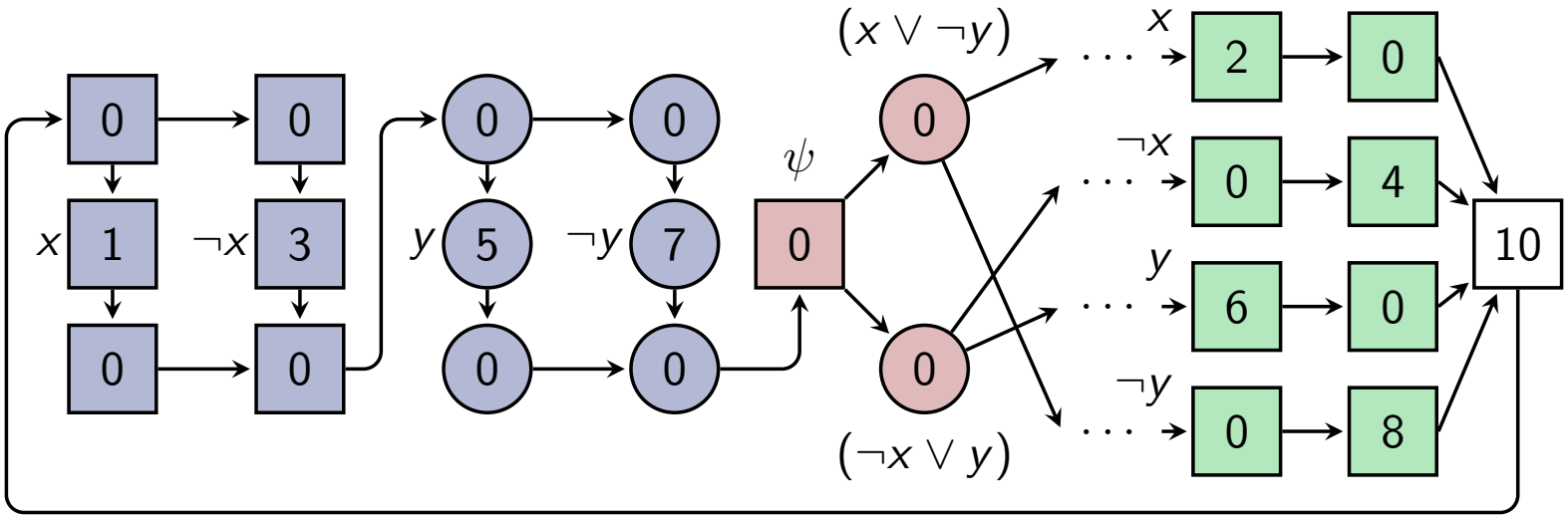


For a well-chosen bound b, a strategy for Player 0 with cost at most b witnesses the truth of φ and vice versa.

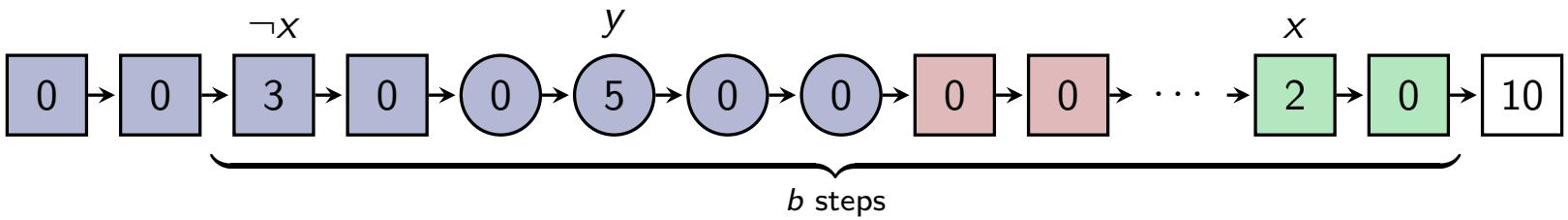


The Reduction

$$\varphi = \forall x \ \exists y \ . \ \overbrace{(x \ \vee \ \neg y) \wedge (\neg x \ \vee \ y)}^{\psi}$$



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



PSPACE-Membership

Lemma (WZ16)

The following problem is in PSPACE*: "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?"*

PSPACE-Membership

Lemma (WZ16)

The following problem is in PSPACE*: "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 Sketch

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

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

PSPACE-Membership

Lemma (WZ16)

The following problem is in PSPACE*: "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 Sketch

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

1. Construct equivalent parity game $\mathcal{G}'$ storing the costs of open requests (up to bound b) and the number of "overflows" (up to bound $|\mathcal{G}|$) $\Rightarrow |\mathcal{G}'| \in |\mathcal{G}|^{\mathcal{O}(d)}$.
2. Define equivalent finite-duration variant $\mathcal{G}'_f$ of $\mathcal{G}'$ with polynomial play-length.
3. $\mathcal{G}'_f$ can be solved on alternating polynomial-time Turing machine.
4. APTIME = PSPACE concludes the proof.

Upper Bounds on Memory

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 $(b+2)^d = 2^{d \log(b+2)}$.

Upper Bounds on Memory

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 $(b+2)^d = 2^{d \log(b+2)}$.

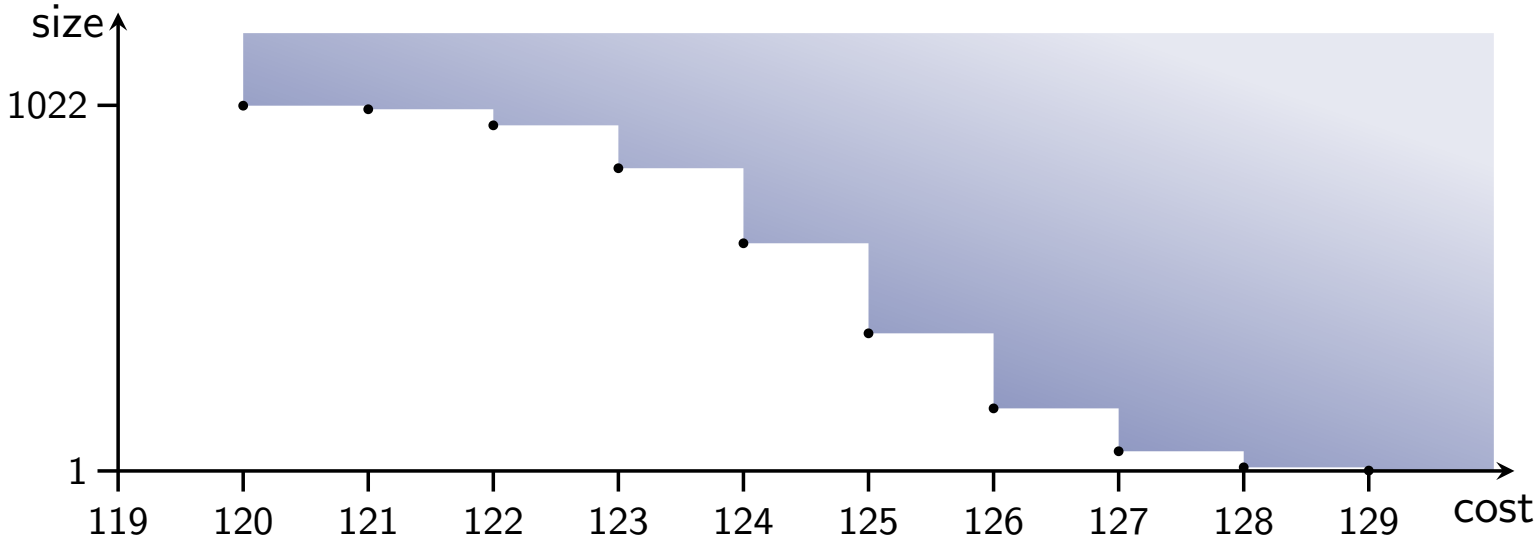
- Recall: lower bound 2^{d-1}.
- The same bounds hold for Player 1.

Tradeoffs

Theorem (WZ16)

Fix some finitary parity game $\mathcal{G}_d$ as before. For every i with $1 \leq i \leq d$ there exists a strategy σ_i for Player 0 in $\mathcal{G}_d$ such that σ_i has cost $d^2 + 3d - i$ and size $\sum_{j=1}^{i-1} \binom{d}{j}$.

Also, every strategy σ' for Player 0 in $\mathcal{G}_d$ whose cost is at most the cost of σ_i has at least the size of σ_i.



Generalizations

We generalized finitary parity games to

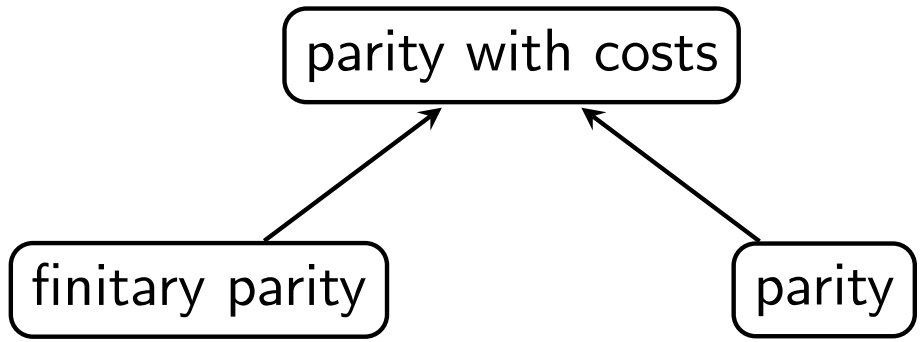
finitary parity

parity

Generalizations

We generalized finitary parity games to

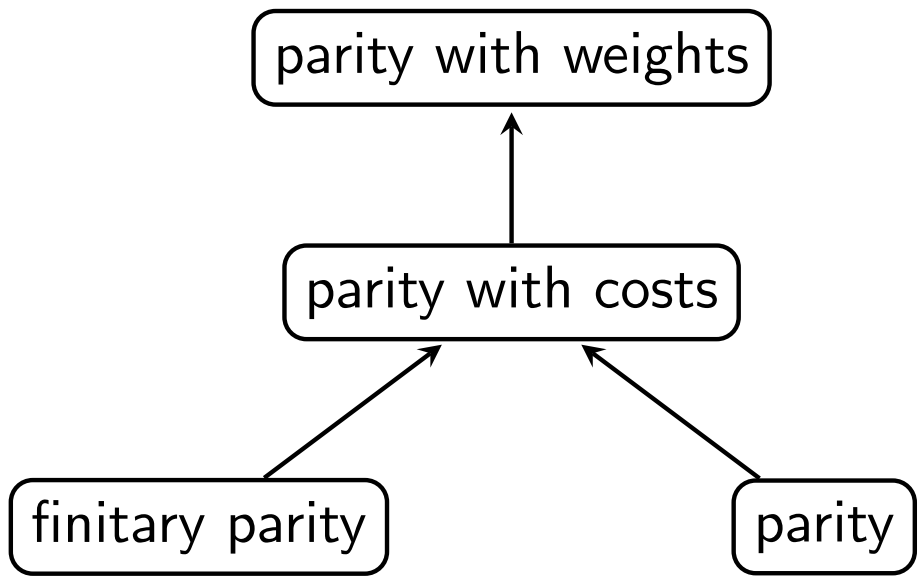
- parity games with costs (by allowing non-negative weights on the edges) **[FZ14]**, and



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]**.



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

Condition	Complexity	Memory Pl. 0	Memory Pl. 1
Parity	quasi-poly	Memoryless	Memoryless
Finitary Parity	PTime	Memoryless	Infinite
Parity w. Costs	quasi-poly	Memoryless	Infinite
Parity w. Weights	$\mathrm{NP} \cap \mathrm{co\text{-}NP}$	Exponential	Infinite

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

Condition	Complexity	Memory Pl. 0 & 1
Finitary Parity	PSPACE-complete	Exponential
Parity w. Costs	PSPACE-complete	Exponential
Parity w. Weights	PSPACE-hard	$\geq$ Exponential

- 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

Introducing LTL by Examples

Atomic propositions $\mathtt{r}_i$ for requests and $\mathtt{g}_i$ for grants.

1. Answer every request: $\bigwedge_i \mathbf{G}\,(\mathtt{r}_i \to \mathbf{F}\,\mathtt{g}_i)$

Introducing LTL by Examples

Atomic propositions $\mathtt{r}_i$ for requests and $\mathtt{g}_i$ for grants.

1. Answer every request: $\bigwedge_i \mathbf{G}\,(\mathtt{r}_i \to \mathbf{F}\,\mathtt{g}_i)$
2. At most one grant at a time: $\mathbf{G} \bigwedge_{i \neq j} \neg(\mathtt{g}_i \wedge \mathtt{g}_j)$

Introducing LTL by Examples

Atomic propositions $\mathtt{r}_i$ for requests and $\mathtt{g}_i$ for grants.

1. Answer every request: $\bigwedge_i \mathbf{G}\,(\mathtt{r}_i \rightarrow \mathbf{F}\,\mathtt{g}_i)$
2. At most one grant at a time: $\mathbf{G} \bigwedge_{i \neq j} \neg(\mathtt{g}_i \wedge \mathtt{g}_j)$
3. No spurious grants:

$$\bigwedge_i \neg[\,(\neg \mathtt{r}_i\,\mathbf{U}\,(\neg \mathtt{r}_i \wedge \mathtt{g}_i))\,] \wedge \neg[\,\mathbf{F}\,(\mathtt{g}_i \wedge \mathbf{X}\,(\neg \mathtt{r}_i\,\mathbf{U}\,(\neg \mathtt{r}_i \wedge \mathtt{g}_i)))\,]$$

Introducing LTL by Examples

Atomic propositions $\mathtt{r}_i$ for requests and $\mathtt{g}_i$ for grants.

1. Answer every request: $\bigwedge_i \mathbf{G}\,(\mathtt{r}_i \rightarrow \mathbf{F}\,\mathtt{g}_i)$
2. At most one grant at a time: $\mathbf{G} \bigwedge_{i \neq j} \neg(\mathtt{g}_i \wedge \mathtt{g}_j)$
3. No spurious grants:

$$\bigwedge_i \neg[\,(\neg \mathtt{r}_i\,\mathbf{U}\,(\neg \mathtt{r}_i \wedge \mathtt{g}_i))\,] \wedge \neg[\,\mathbf{F}\,(\mathtt{g}_i \wedge \mathbf{X}\,(\neg \mathtt{r}_i\,\mathbf{U}\,(\neg \mathtt{r}_i \wedge \mathtt{g}_i)))\,]$$

$$\equiv \bigwedge_i [\,(\mathtt{r}_i\,\mathbf{R}\,(\mathtt{r}_i \vee \neg \mathtt{g}_i))\,] \wedge [\,\mathbf{G}\,(\neg \mathtt{g}_i \vee \mathbf{X}\,(\mathtt{r}_i\,\mathbf{R}\,(\mathtt{r}_i \vee \neg \mathtt{g}_i)))\,]$$

A Problem with LTL

- Answer every request: $\bigwedge_i \mathbf{G}\,(\mathtt{r}_i \to \mathbf{F}\,\mathtt{g}_i)$

$$\mathtt{r}_0 \to \mathtt{g}_0 \to \mathtt{r}_0 \longrightarrow \quad \longrightarrow \mathtt{g}_0 \to \mathtt{r}_0 \longrightarrow \quad \longrightarrow \quad \longrightarrow \mathtt{g}_0 \to \cdots$$

A Problem with LTL

- Answer every request: $\bigwedge_i \mathbf{G}\,(\mathtt{r}_i \to \mathbf{F}\,\mathtt{g}_i)$

$$\mathtt{r}_0 \to \mathtt{g}_0 \to \mathtt{r}_0 \longrightarrow \quad \longrightarrow \mathtt{g}_0 \to \mathtt{r}_0 \longrightarrow \quad \longrightarrow \quad \longrightarrow \mathtt{g}_0 \to \cdots$$

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

LTL goes Quantitative

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

Syntax: Add prompt-eventually operator $\mathbf{F_P}$.

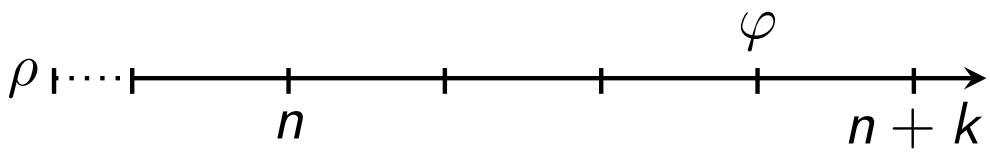
$$\varphi ::= p \mid \neg p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \mathbf{X}\,\varphi \mid \varphi\,\mathbf{U}\,\varphi \mid \varphi\,\mathbf{R}\,\varphi \mid \mathbf{F_P}\,\varphi$$

Prompt-LTL

Syntax: Add prompt-eventually operator $\mathbf{F_P}$.

$$\varphi ::= p \mid \neg p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \mathbf{X}\,\varphi \mid \varphi\,\mathbf{U}\,\varphi \mid \varphi\,\mathbf{R}\,\varphi \mid \mathbf{F_P}\,\varphi$$

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

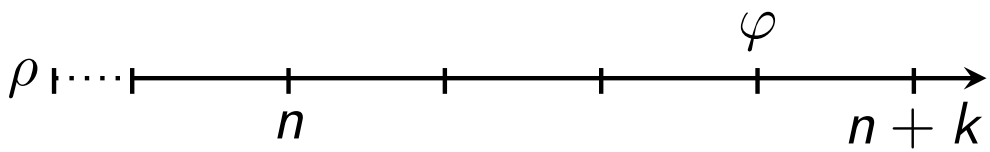
- $(\rho, n, k) \models \mathbf{F_P}\,\varphi$: 

Prompt-LTL

Syntax: Add prompt-eventually operator $\mathbf{F_P}$.

$$\varphi ::= p \mid \neg p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \mathbf{X}\,\varphi \mid \varphi\,\mathbf{U}\,\varphi \mid \varphi\,\mathbf{R}\,\varphi \mid \mathbf{F_P}\,\varphi$$

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

- $(\rho, n, k) \models \mathbf{F_P}\,\varphi$: 

Now: $\bigwedge_i \mathbf{G}\,(\mathtt{r}_i \to \mathbf{F_P}\,\mathtt{g}_i)$

Prompt-LTL Games

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

Prompt-LTL Games

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

$\mathrm{PROMPT\text{–}LTL}$ games are not harder than LTL games...

Theorem (KPV07)

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

Prompt-LTL Games

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

...unless you optimize the bound.

Theorem (Z11)

1. *The* $\mathrm{PROMPT\text{–}LTL}$ *game optimization problem can be solved in triply-exponential time.*
2. *The bound* k_φ *is tight in general.*

Prompt-LTL Games

- Label the arena by atomic propositions.
- Winning condition: $\mathrm{PROMPT\text{–}LTL}$ formula φ.
- Player 0 wins if there is a uniform bound k and a strategy σ such that every play that is consistent with σ satisfies the winning condition φ 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* $\mathcal{G}$ *with winning condition* φ *and* $k \leq k_\varphi$. *There is an* LTL *game* $\mathcal{G}_k$ *such that*

1. *if Player* 0 *wins* $\mathcal{G}$ *w.r.t.* k, *then she wins* $\mathcal{G}_k$,
2. *if Player* 0 *wins* $\mathcal{G}_k$, *then she wins* $\mathcal{G}$ *w.r.t.* $2k$, *and*
3. $\mathcal{G}_k$ *can be solved in doubly-exponential time in* $|\mathcal{G}|$.

An Approximation Algorithm

Lemma (TWZ16)

Fix a PROMPT–LTL *game* $\mathcal{G}$ *with winning condition* φ *and* $k \leq k_\varphi$. *There is an* LTL *game* $\mathcal{G}_k$ *such that*

1. *if Player* 0 *wins* $\mathcal{G}$ *w.r.t.* k, *then she wins* $\mathcal{G}_k$,
2. *if Player* 0 *wins* $\mathcal{G}_k$, *then she wins* $\mathcal{G}$ *w.r.t.* $2k$, *and*
3. $\mathcal{G}_k$ *can be solved in doubly-exponential time in* $|\mathcal{G}|$.

The algorithm:

1: **for** $k = 0;\ k \leq k_\varphi;\ k \leftarrow k + 1$ **do**
2: **if** Player 0 wins $\mathcal{G}_k$ **then**
3: **return** $2k$

An Approximation Algorithm

Lemma (TWZ16)

Fix a PROMPT–LTL *game* $\mathcal{G}$ *with winning condition* φ *and* $k \leq k_\varphi$. *There is an* LTL *game* $\mathcal{G}_k$ *such that*

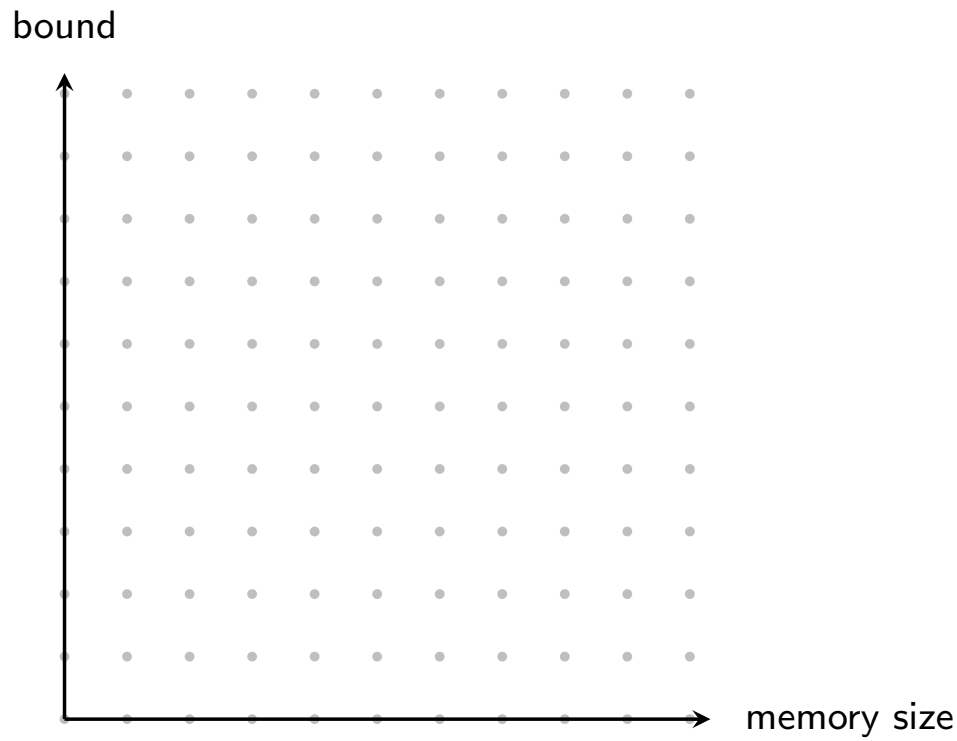
1. *if Player* 0 *wins* $\mathcal{G}$ *w.r.t.* k, *then she wins* $\mathcal{G}_k$,
2. *if Player* 0 *wins* $\mathcal{G}_k$, *then she wins* $\mathcal{G}$ *w.r.t.* $2k$, *and*
3. $\mathcal{G}_k$ *can be solved in doubly-exponential time in* $|\mathcal{G}|$.

The algorithm:

1: **for** $k = 0$; $k \leq k_\varphi$; $k \leftarrow k + 1$ **do**
2: **if** Player 0 wins $\mathcal{G}_k$ **then**
3: **return** $2k$

- Running time: doubly-exponential
- Approximation ratio: 2
- Yields winning strategy

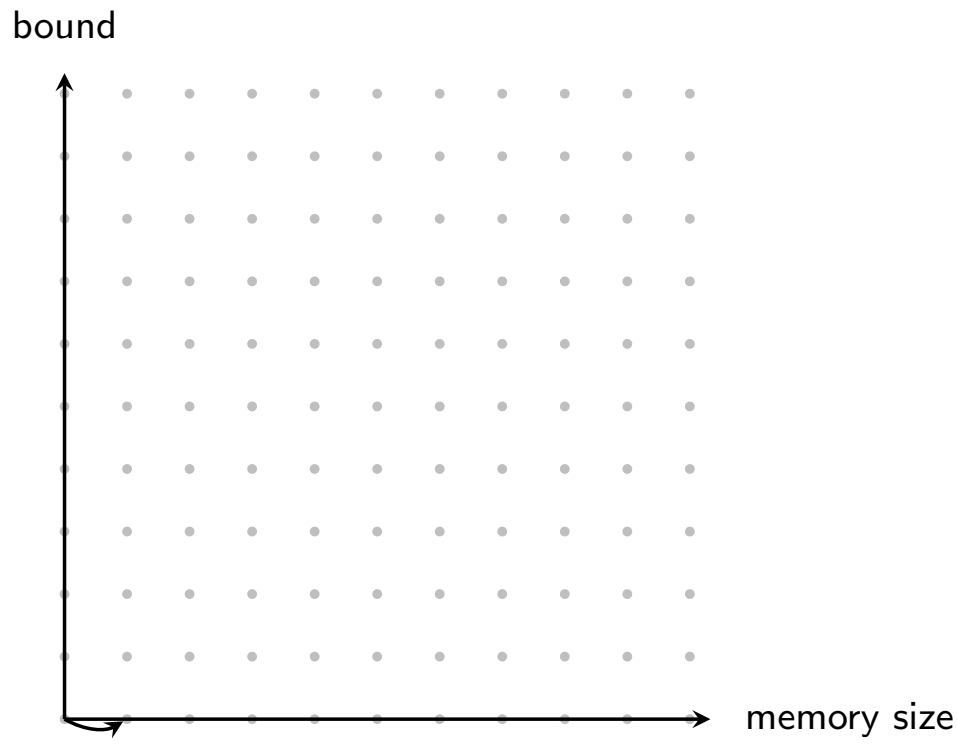
Tradeoffs



Implementation via bounded synthesis:

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

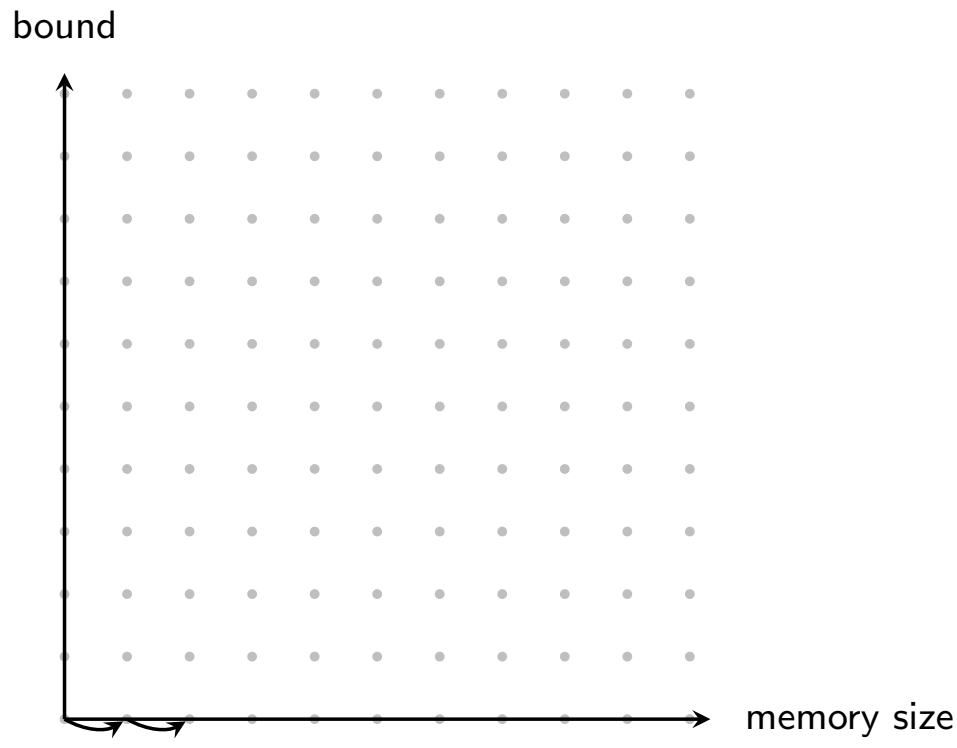
Tradeoffs



Implementation via bounded synthesis:

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

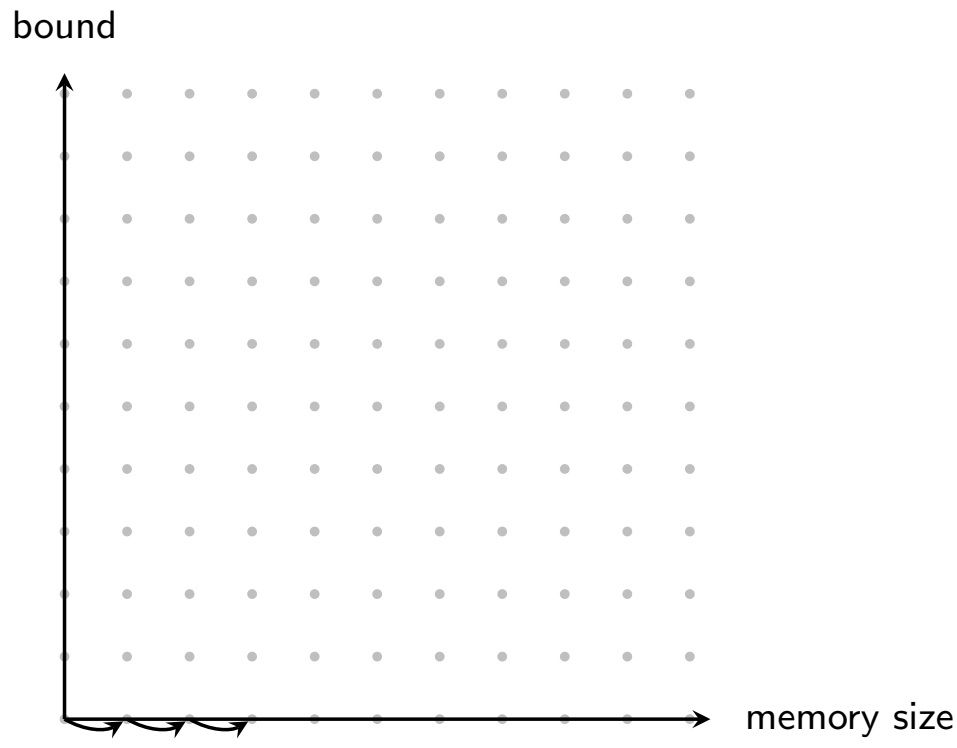
Tradeoffs



Implementation via bounded synthesis:

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

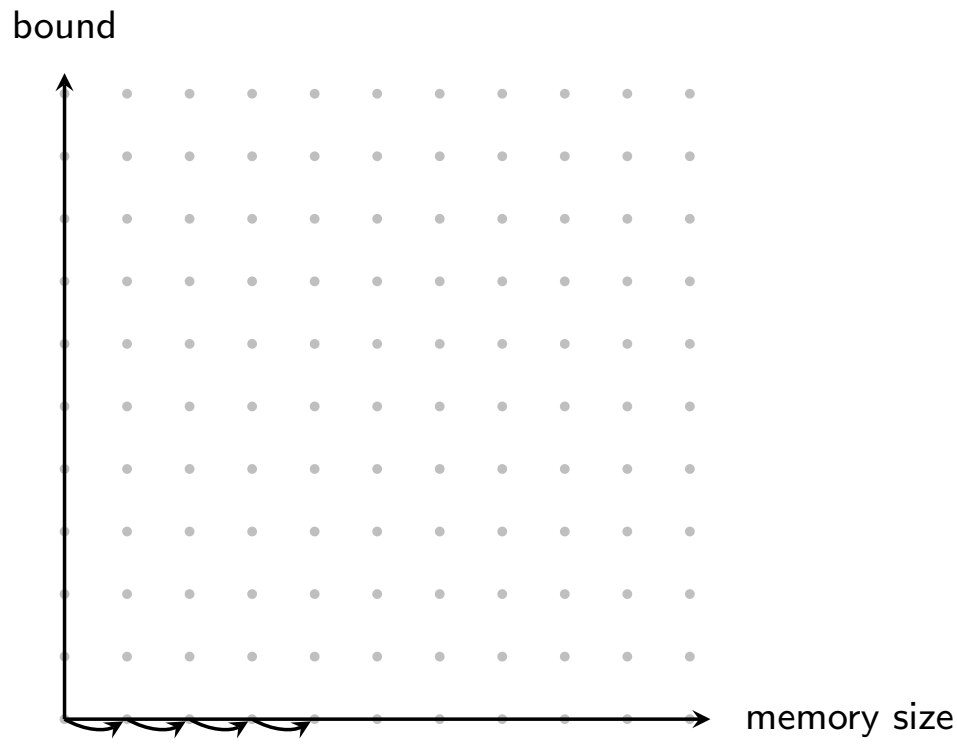
Tradeoffs



Implementation via bounded synthesis:

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

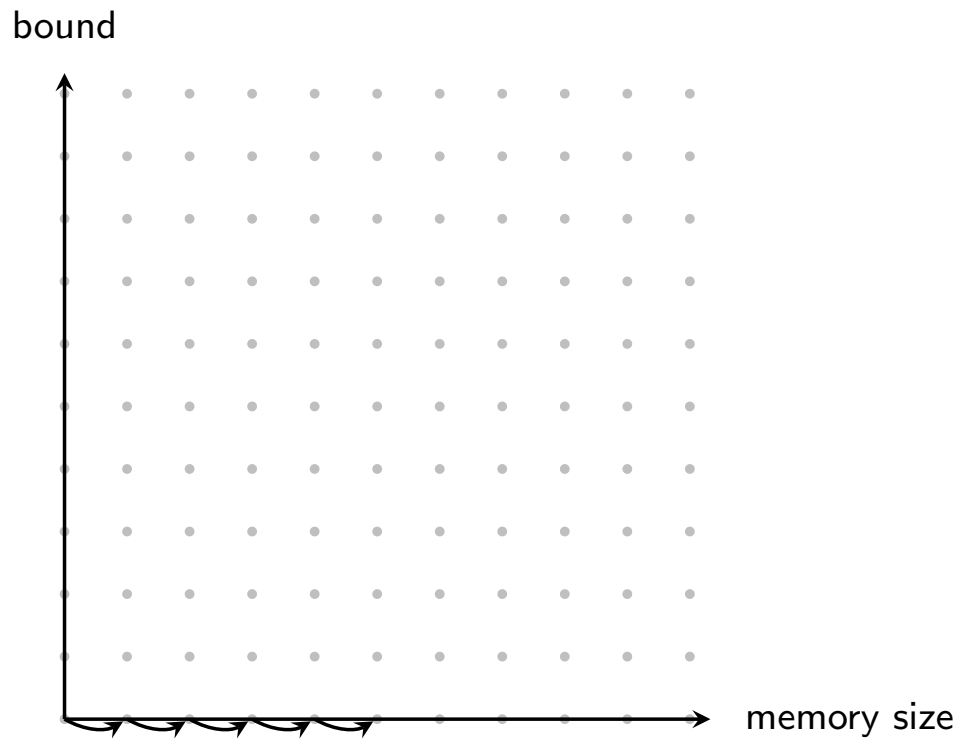
Tradeoffs



Implementation via bounded synthesis:

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

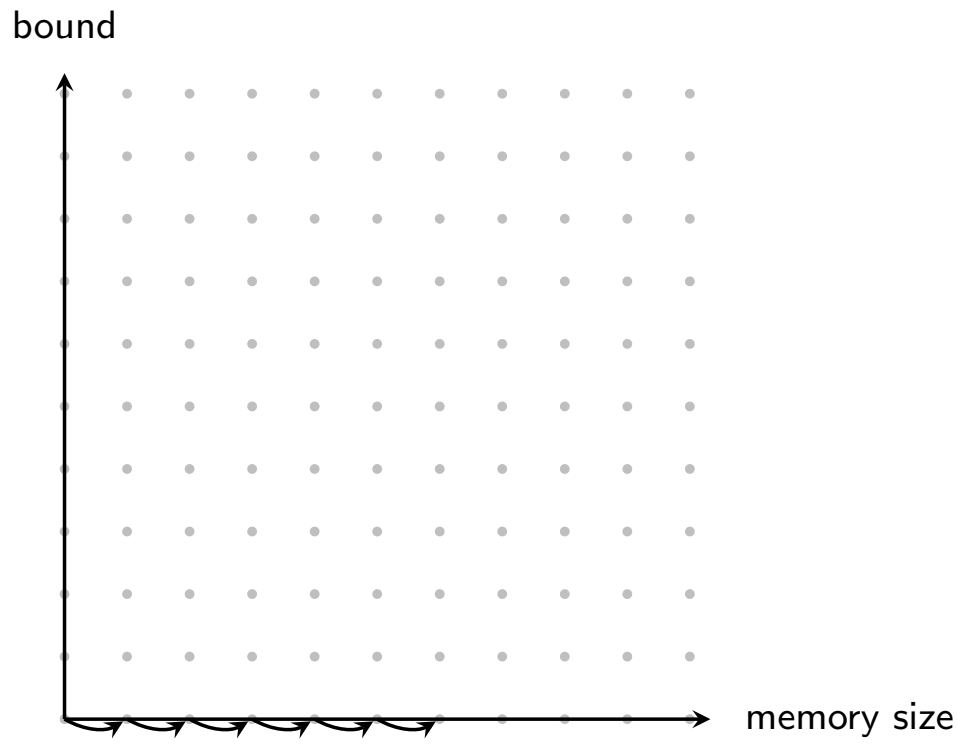
Tradeoffs



Implementation via bounded synthesis:

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

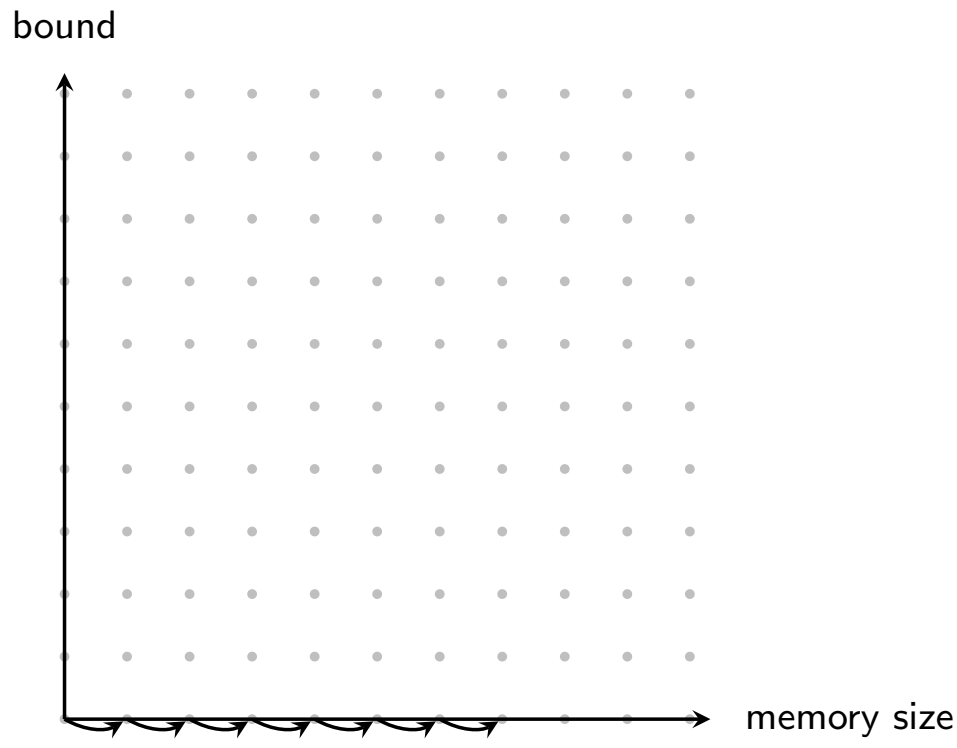
Tradeoffs



Implementation via bounded synthesis:

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

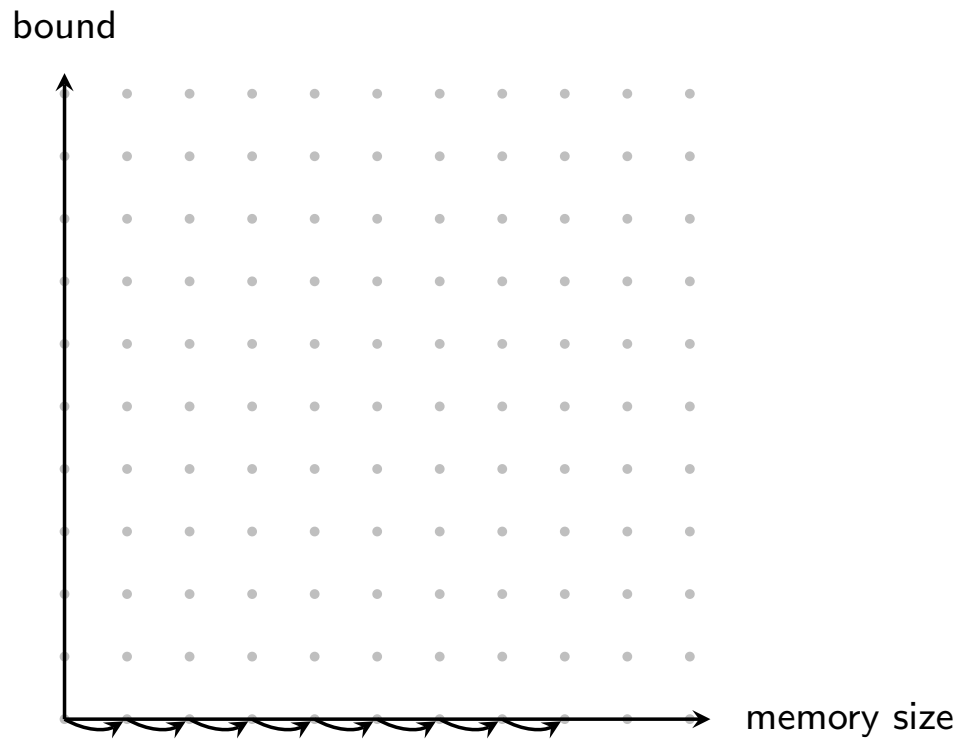
Tradeoffs



Implementation via bounded synthesis:

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

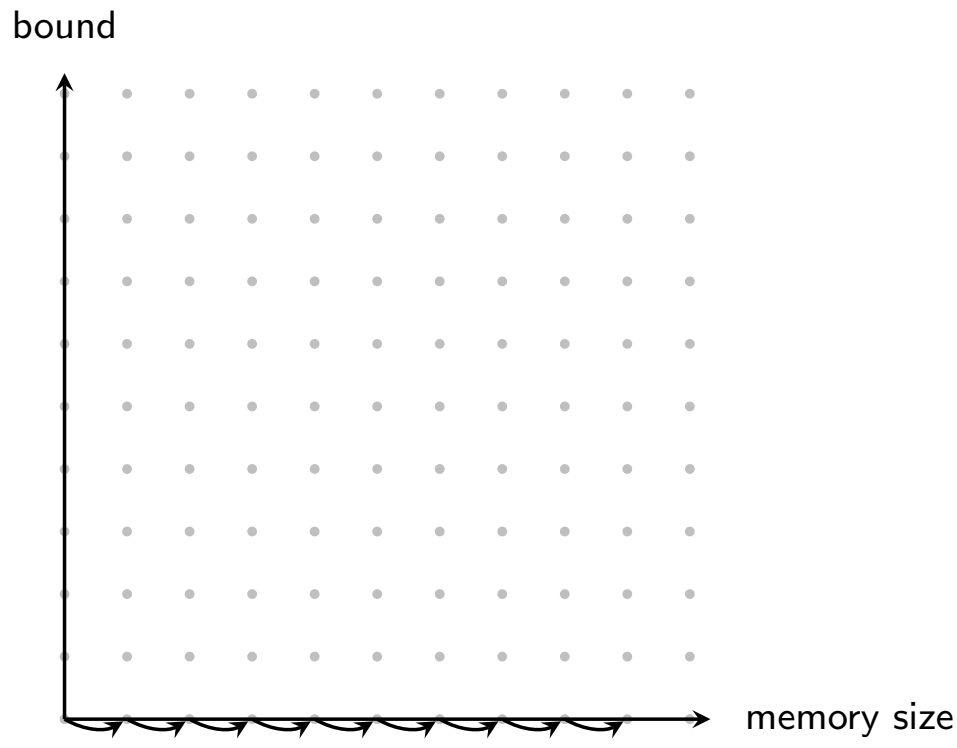
Tradeoffs



Implementation via bounded synthesis:

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

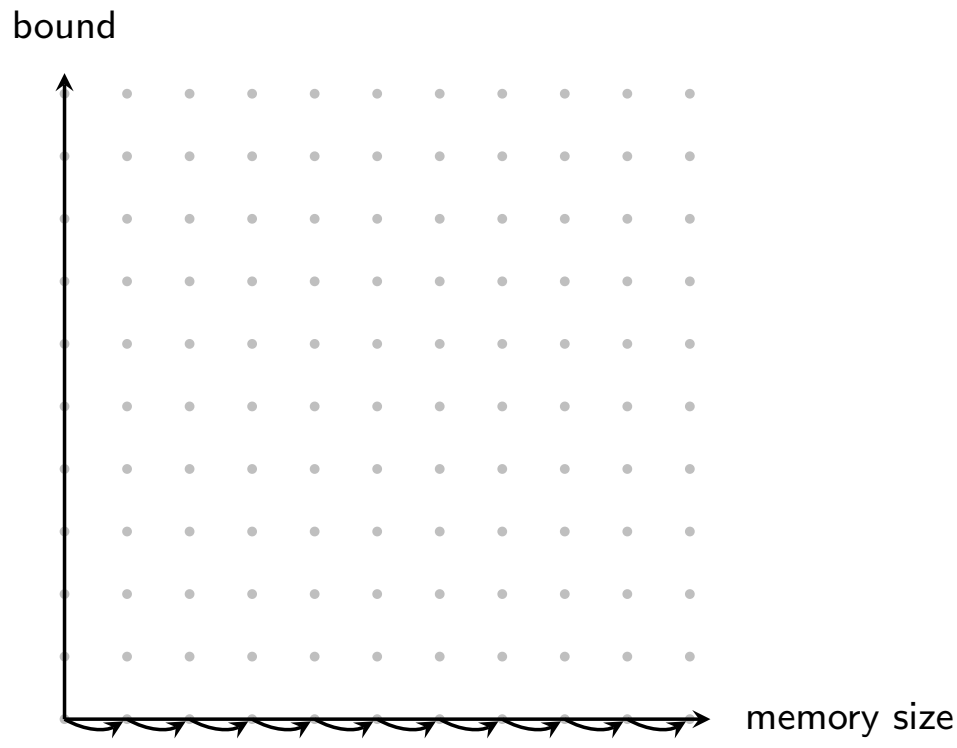
Tradeoffs



Implementation via bounded synthesis:

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

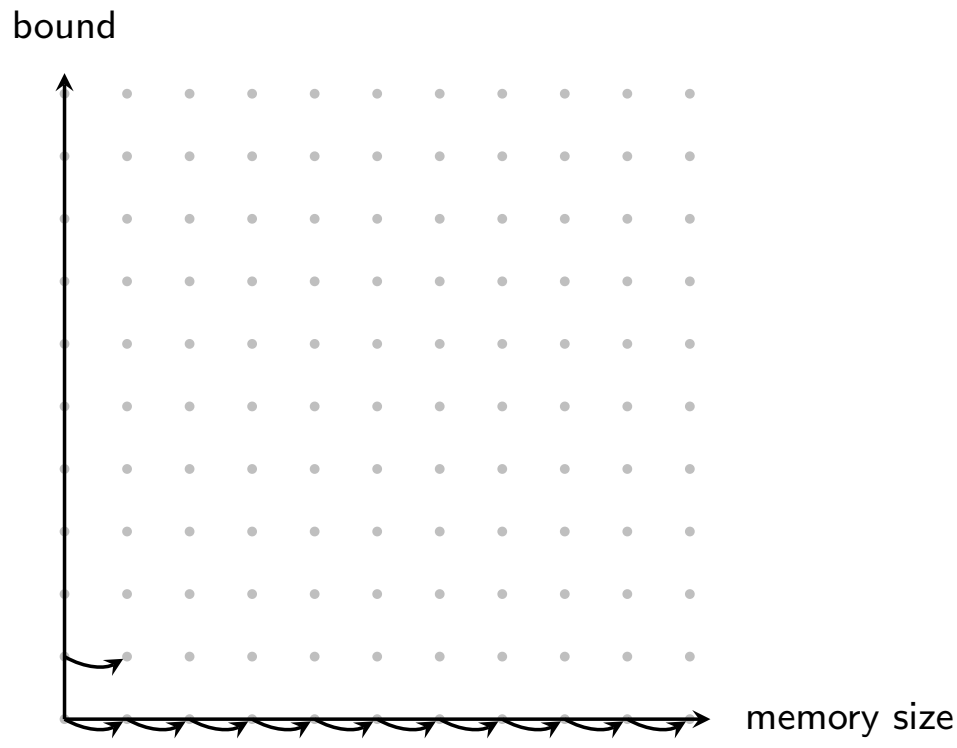
Tradeoffs



Implementation via bounded synthesis:

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

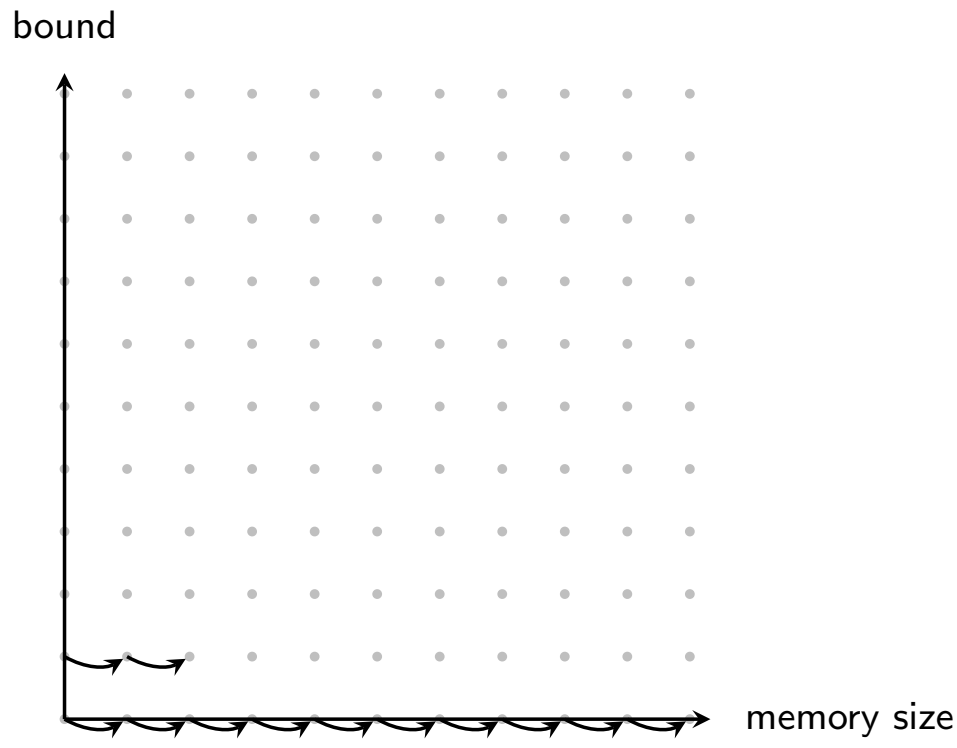
Tradeoffs



Implementation via bounded synthesis:

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

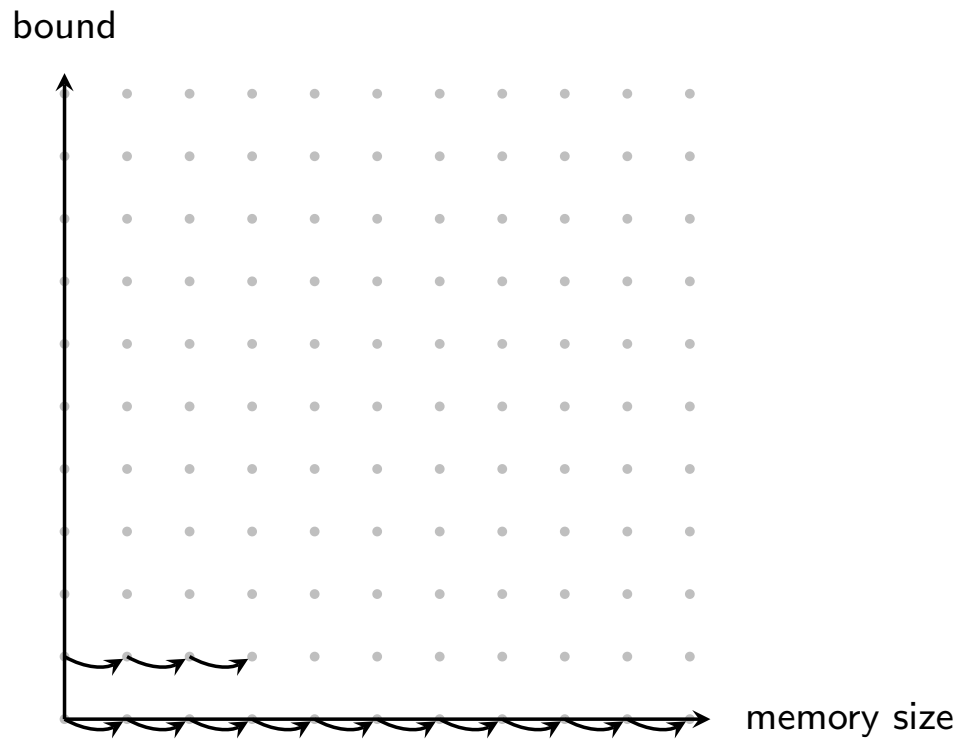
Tradeoffs



Implementation via bounded synthesis:

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

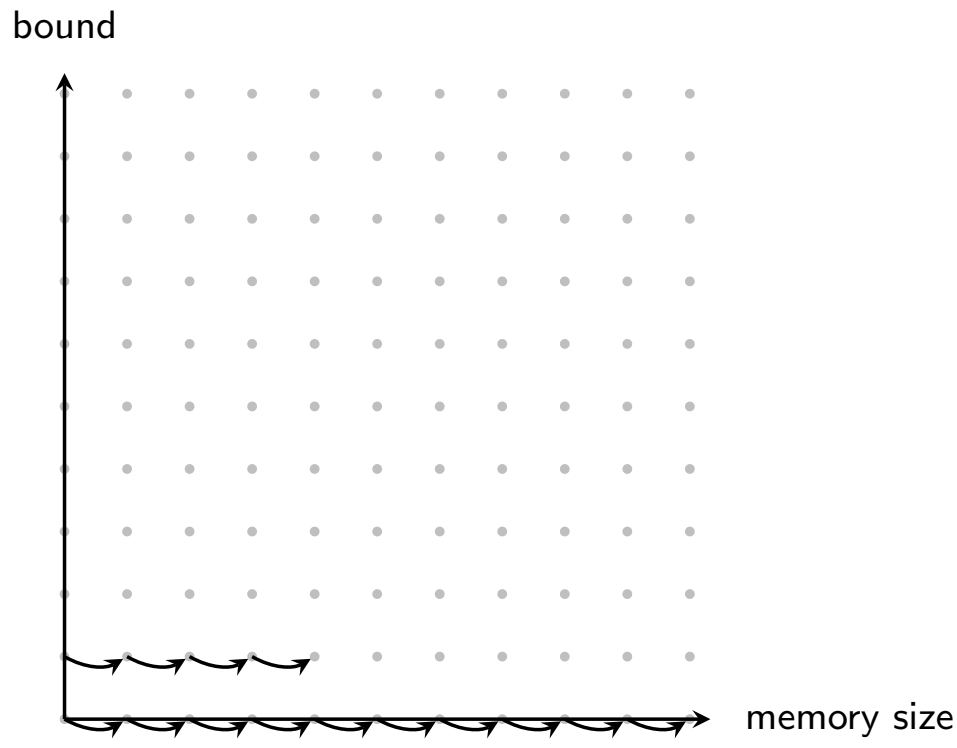
Tradeoffs



Implementation via bounded synthesis:

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

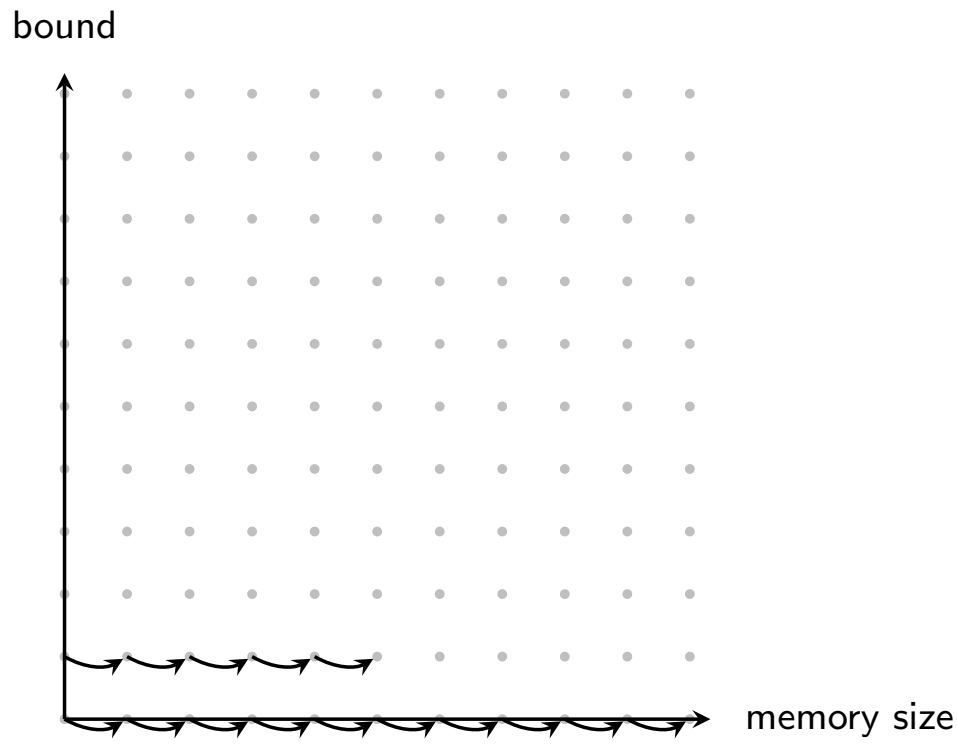
Tradeoffs



Implementation via bounded synthesis:

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

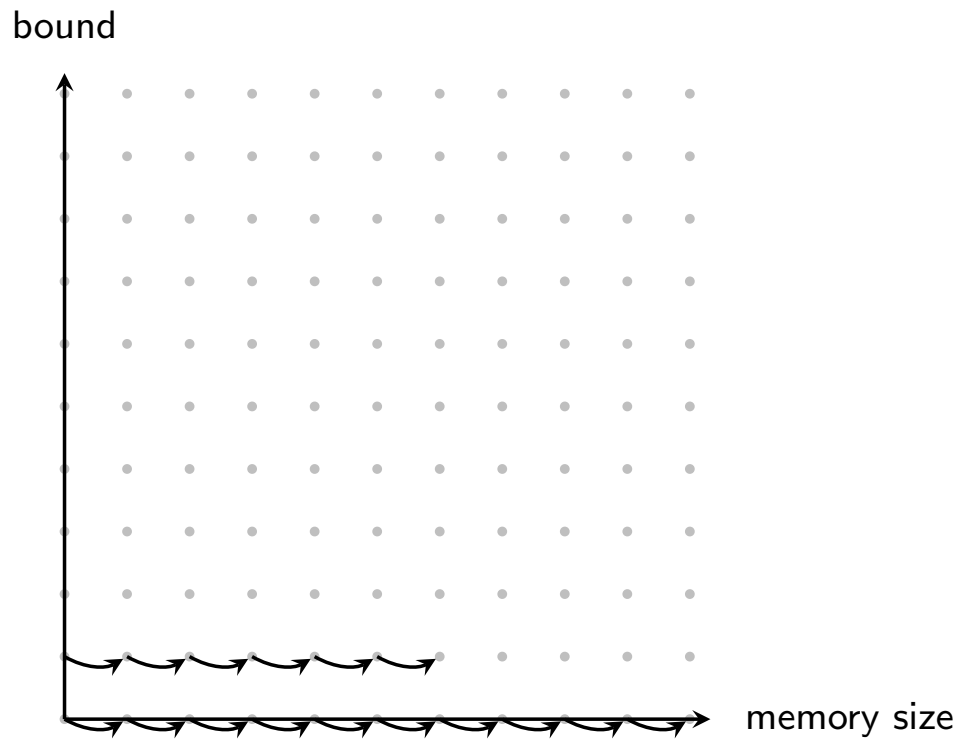
Tradeoffs



Implementation via bounded synthesis:

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

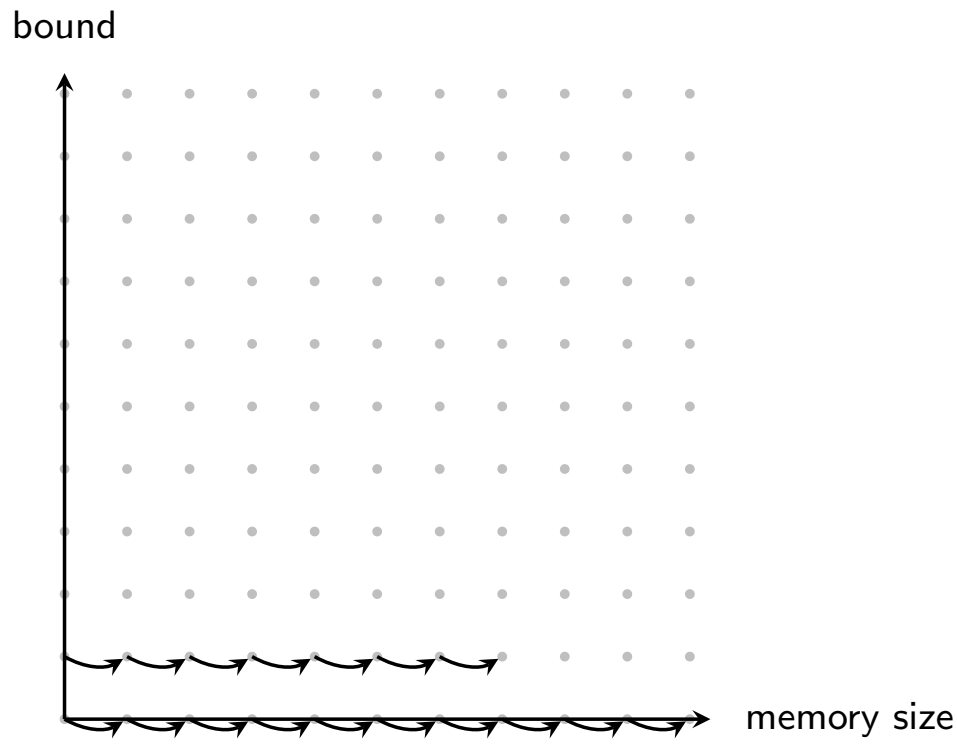
Tradeoffs



Implementation via bounded synthesis:

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

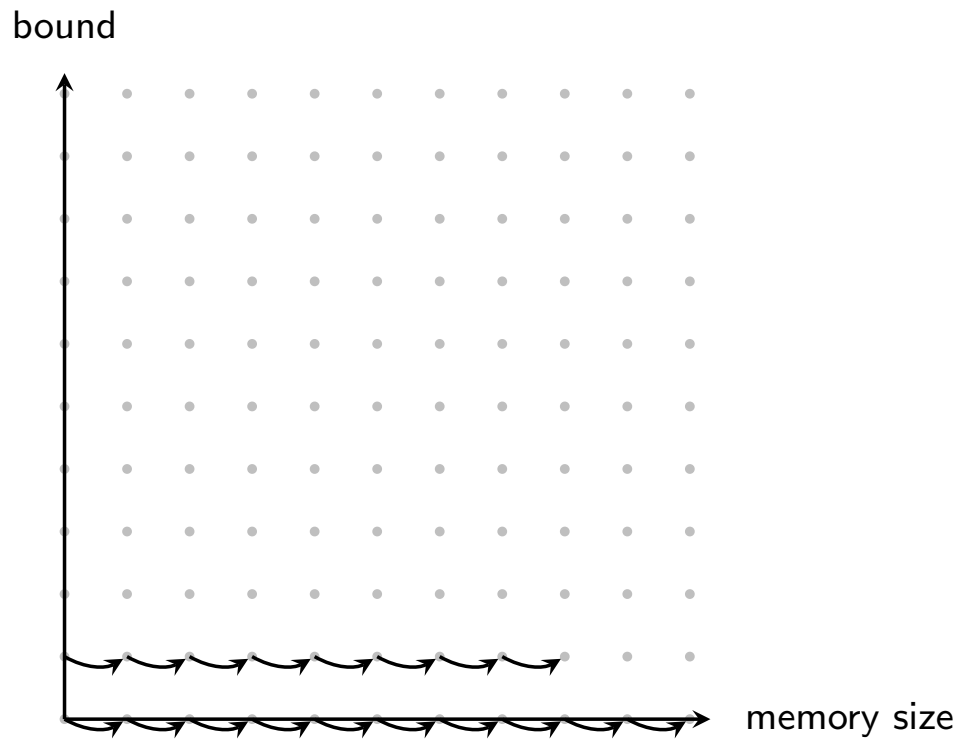
Tradeoffs



Implementation via bounded synthesis:

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

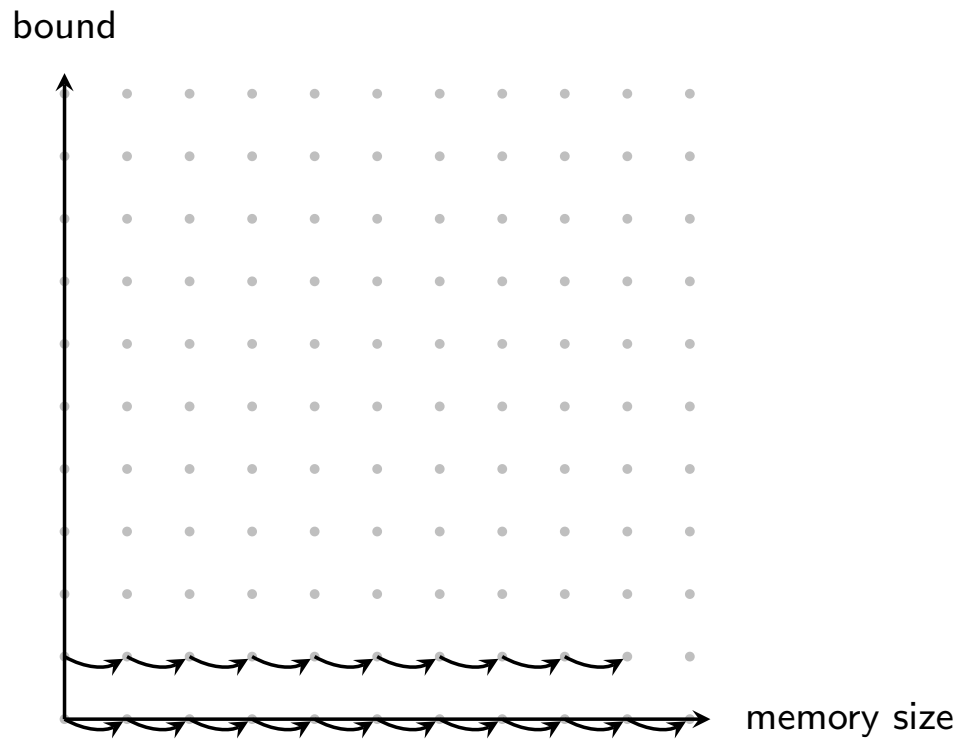
Tradeoffs



Implementation via bounded synthesis:

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

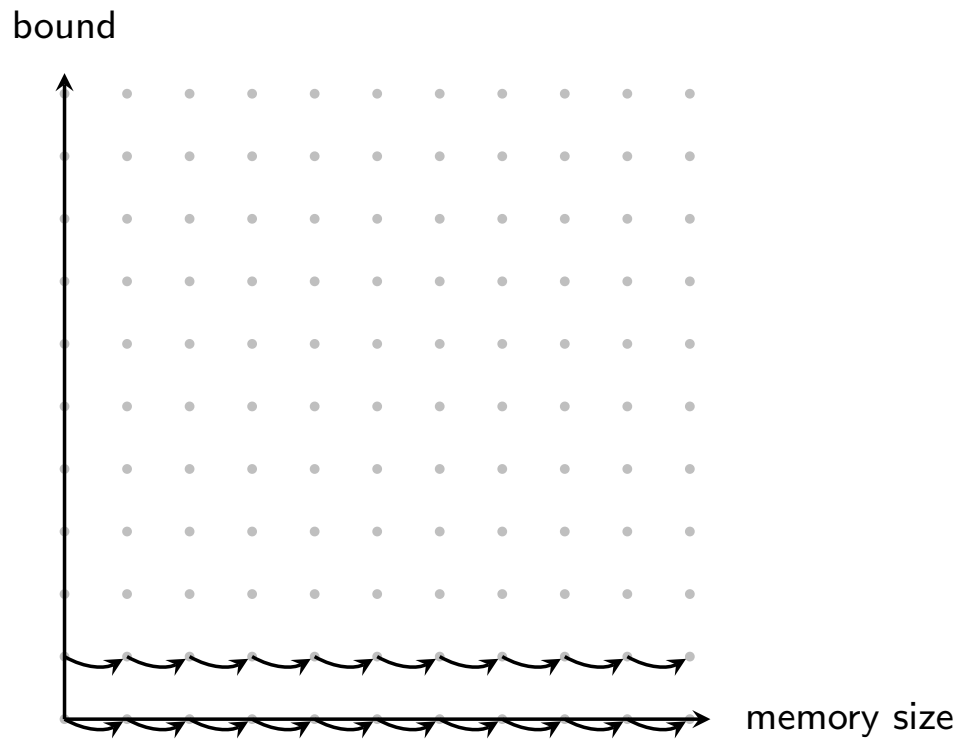
Tradeoffs



Implementation via bounded synthesis:

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

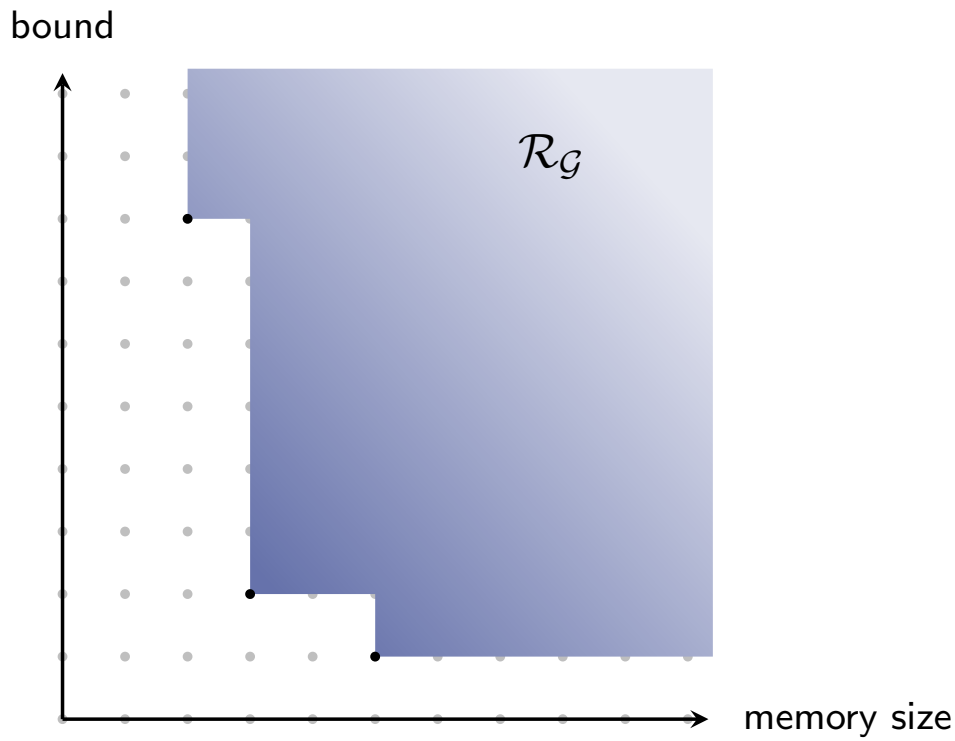
Tradeoffs



Implementation via bounded synthesis:

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

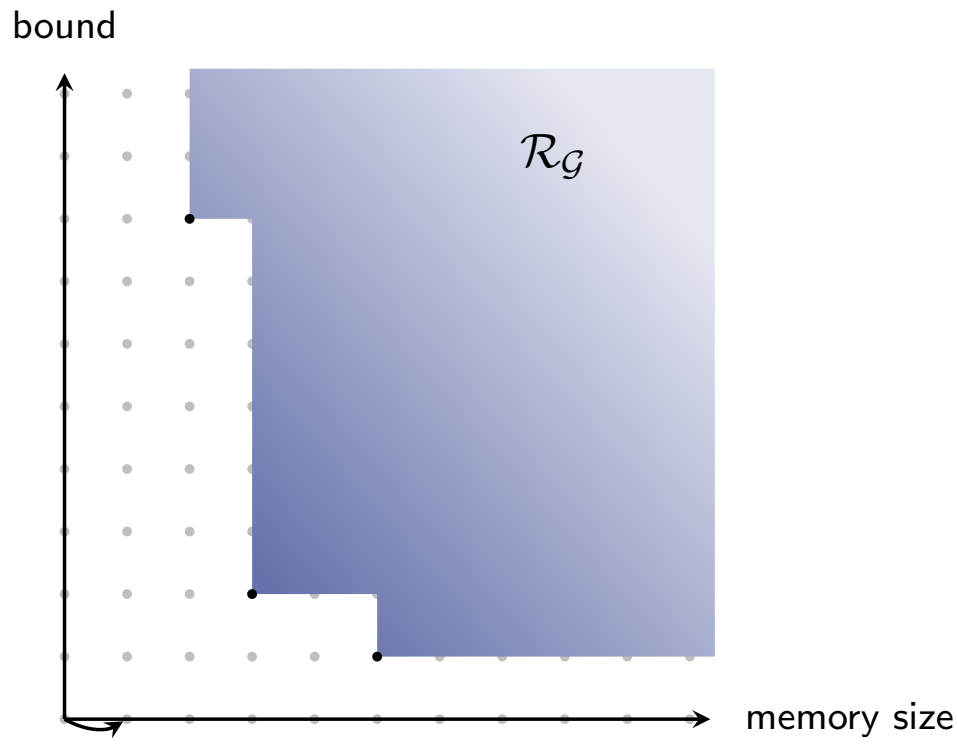
Tradeoffs



Implementation via bounded synthesis:

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

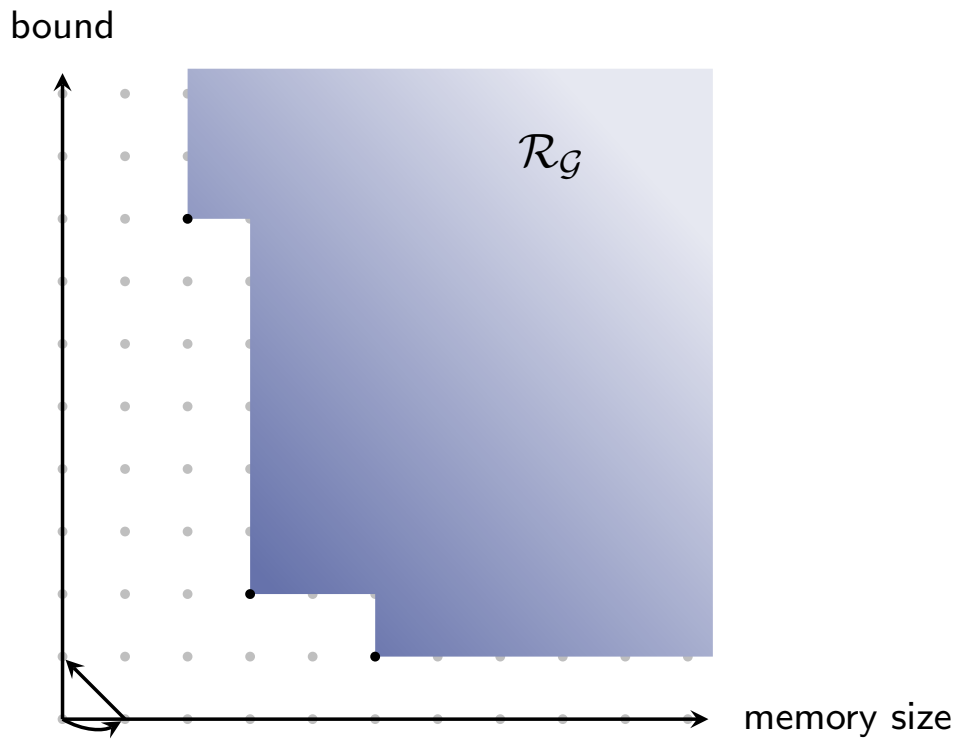
Tradeoffs



Implementation via bounded synthesis:

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

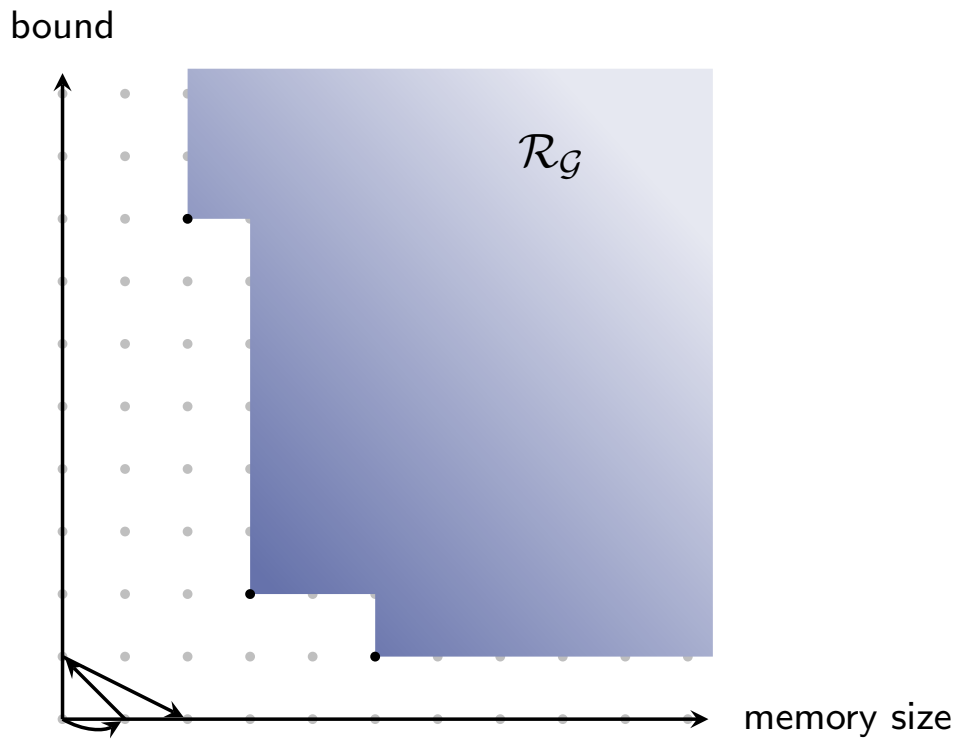
Tradeoffs



Implementation via bounded synthesis:

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

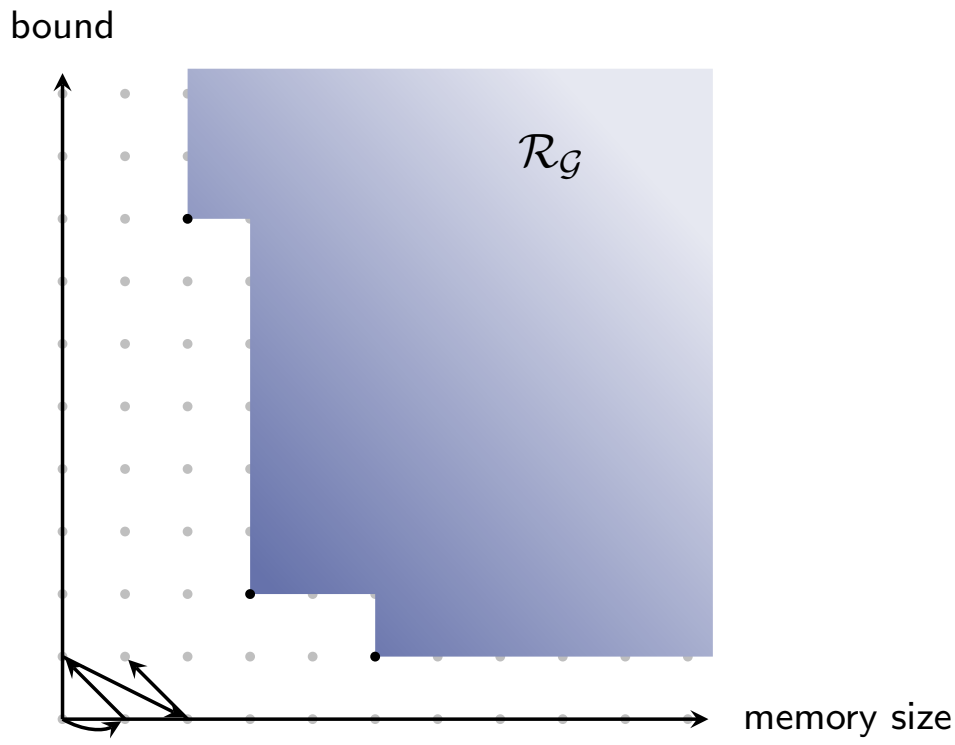
Tradeoffs



Implementation via bounded synthesis:

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

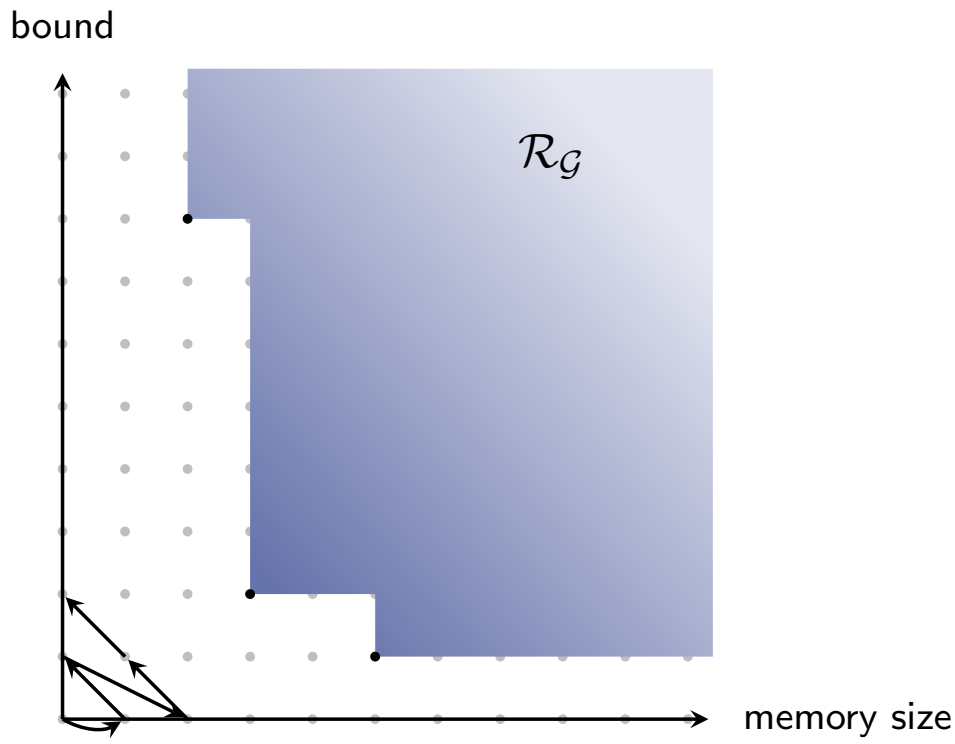
Tradeoffs



Implementation via bounded synthesis:

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

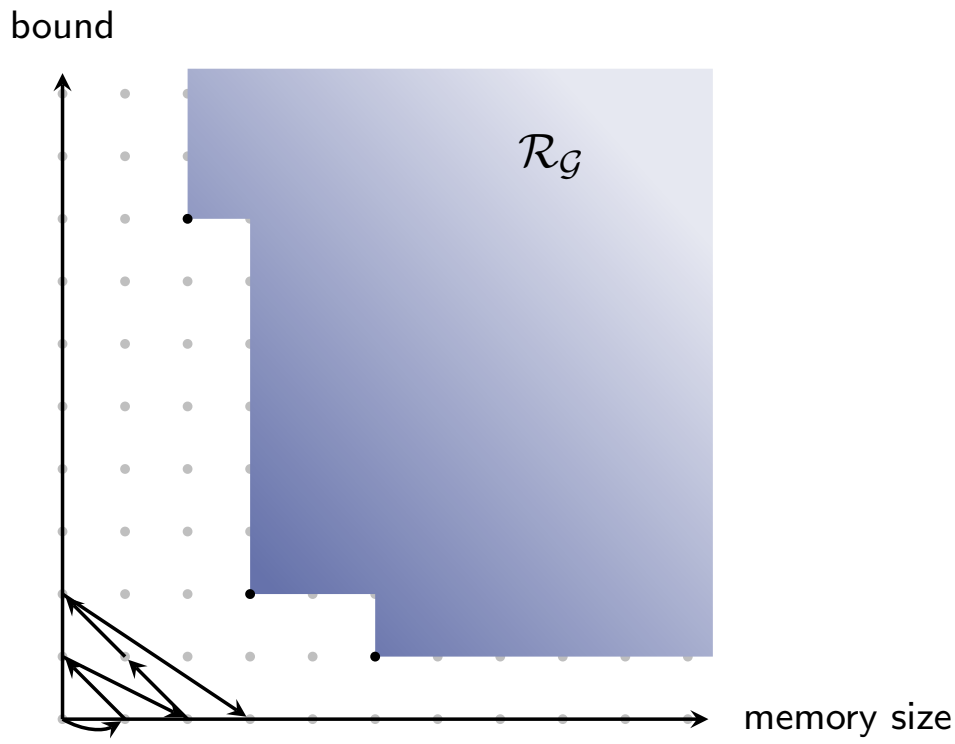
Tradeoffs



Implementation via bounded synthesis:

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

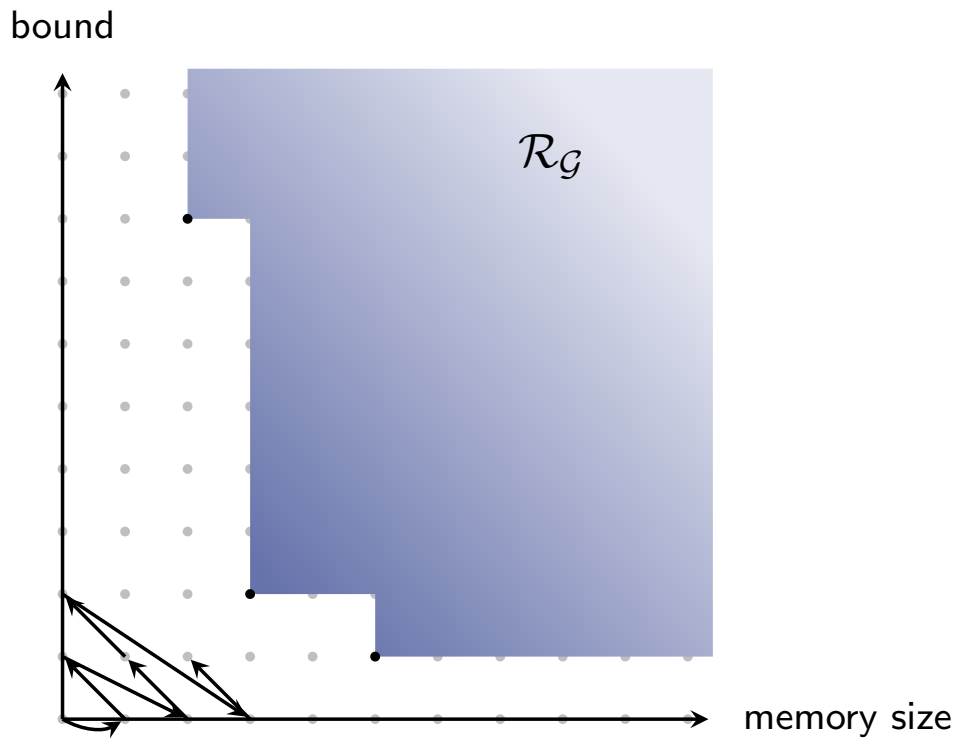
Tradeoffs



Implementation via bounded synthesis:

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

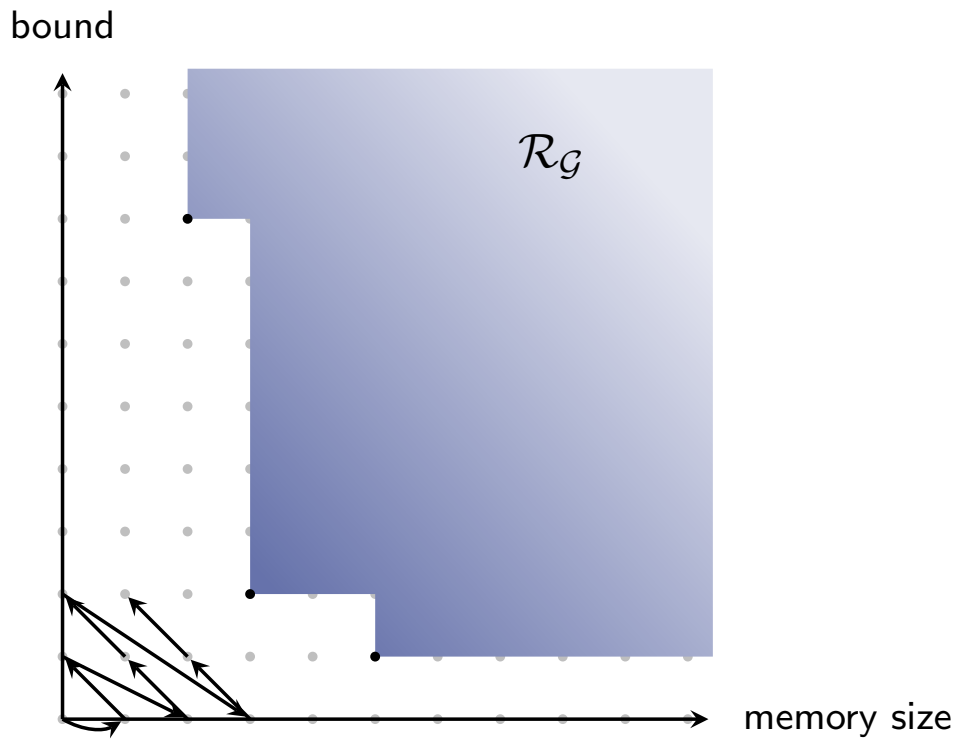
Tradeoffs



Implementation via bounded synthesis:

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

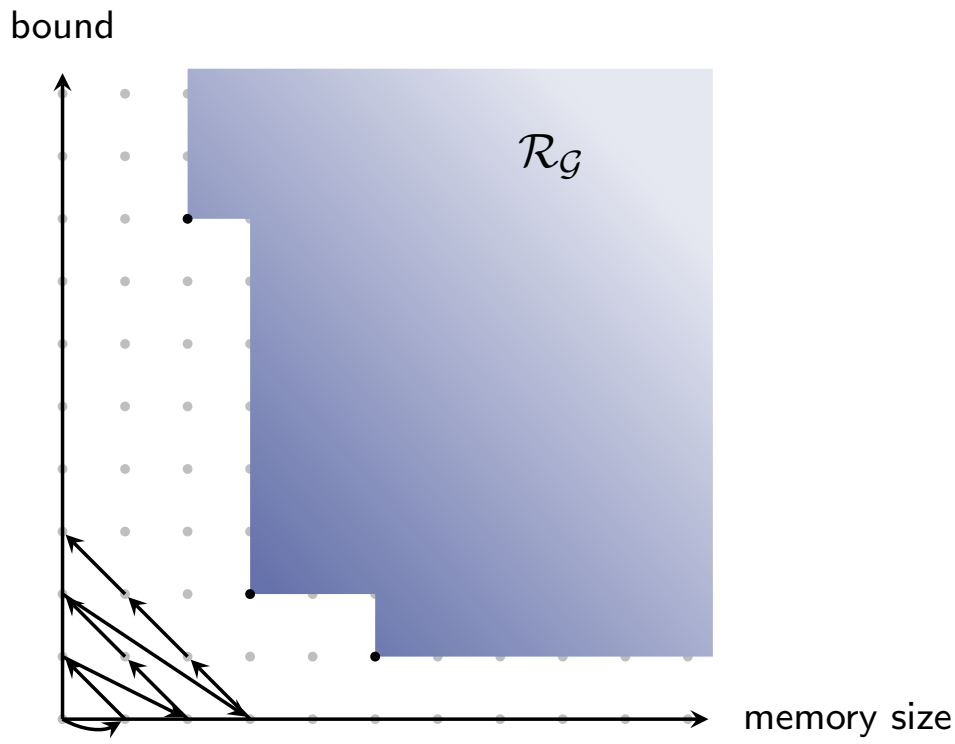
Tradeoffs



Implementation via bounded synthesis:

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

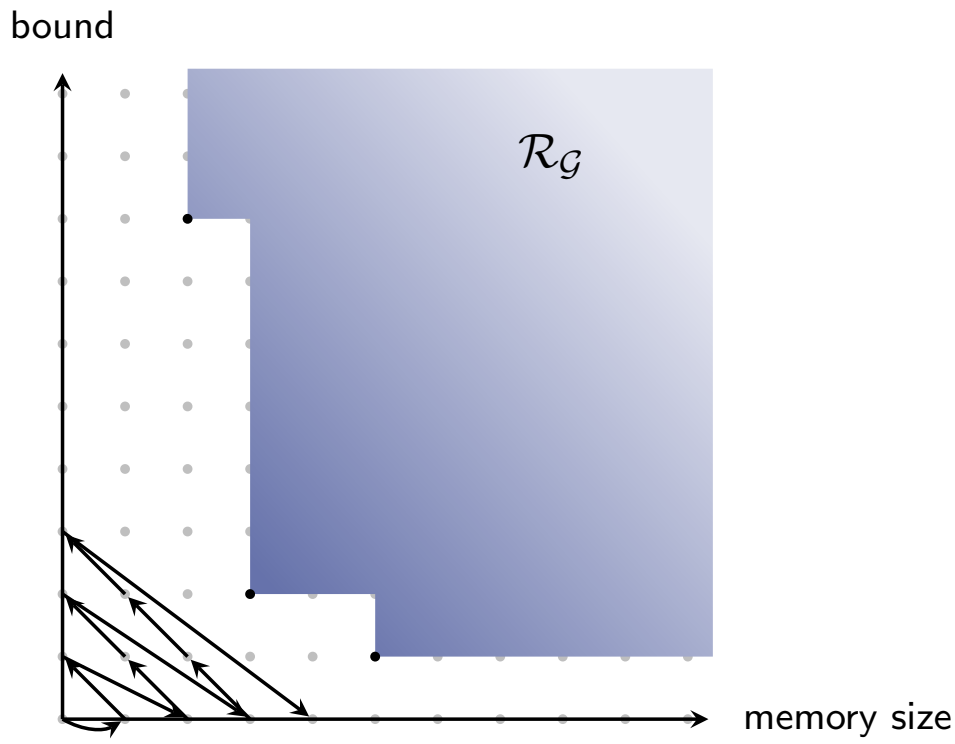
Tradeoffs



Implementation via bounded synthesis:

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

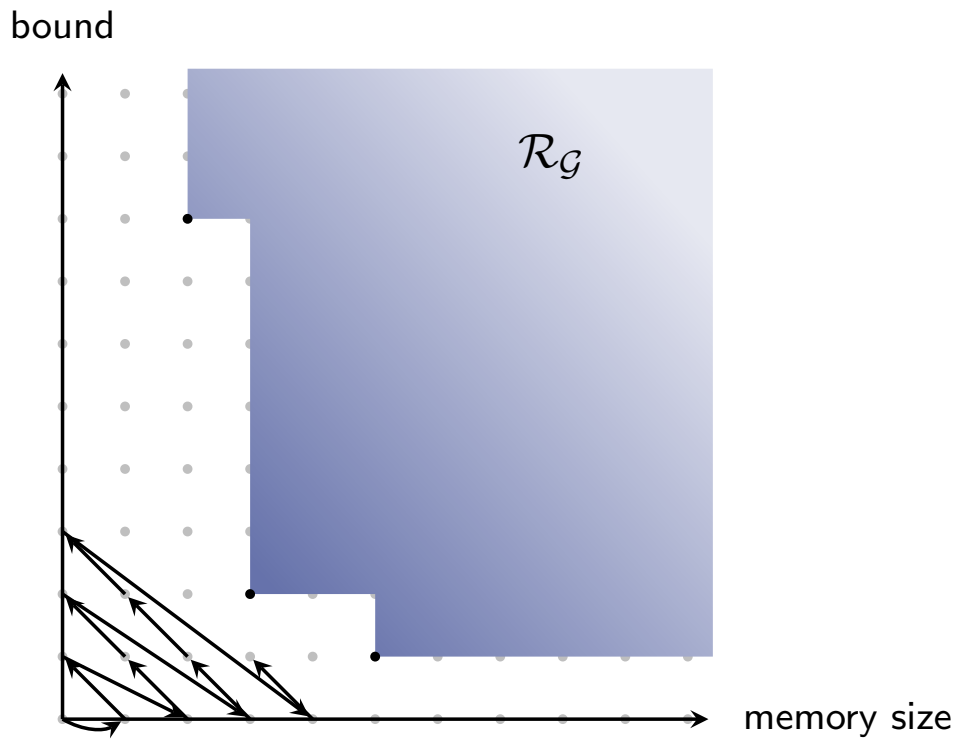
Tradeoffs



Implementation via bounded synthesis:

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

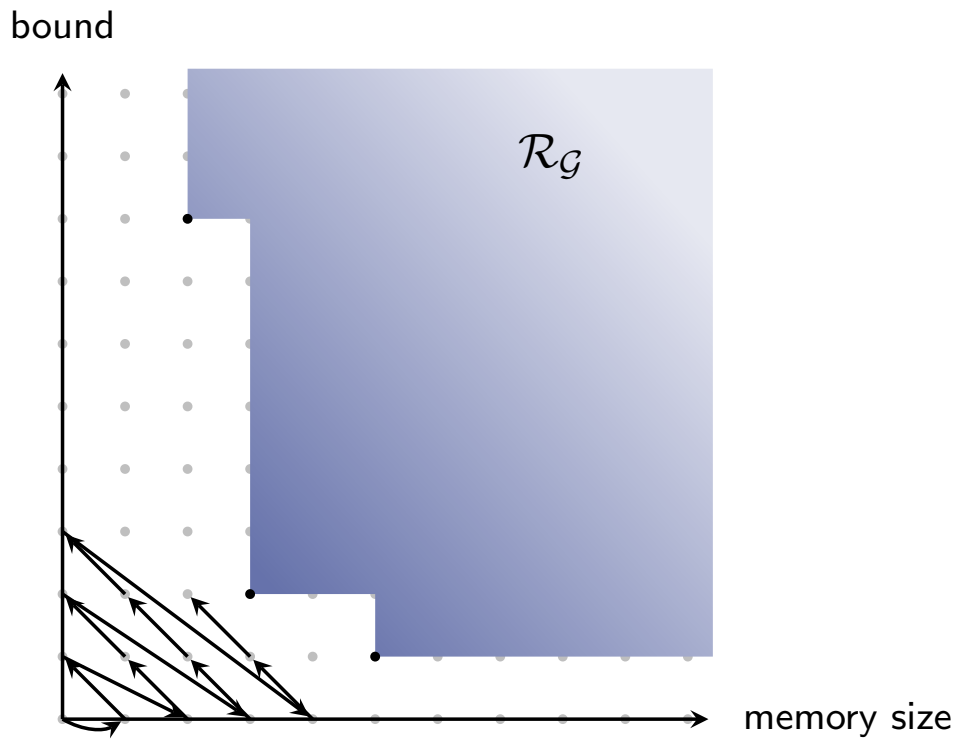
Tradeoffs



Implementation via bounded synthesis:

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

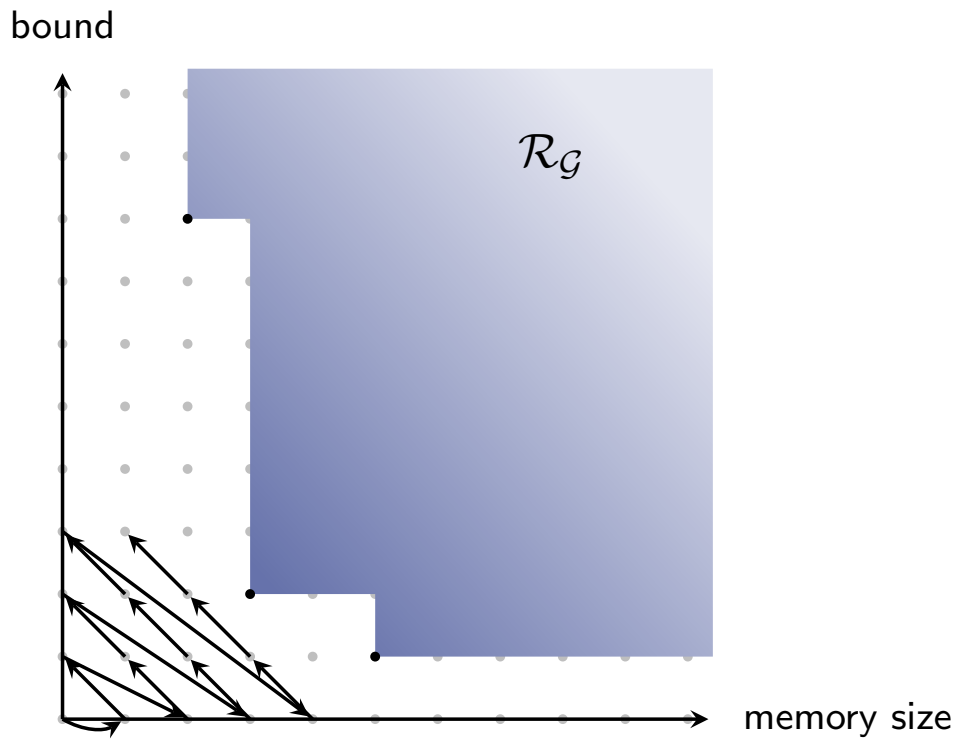
Tradeoffs



Implementation via bounded synthesis:

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

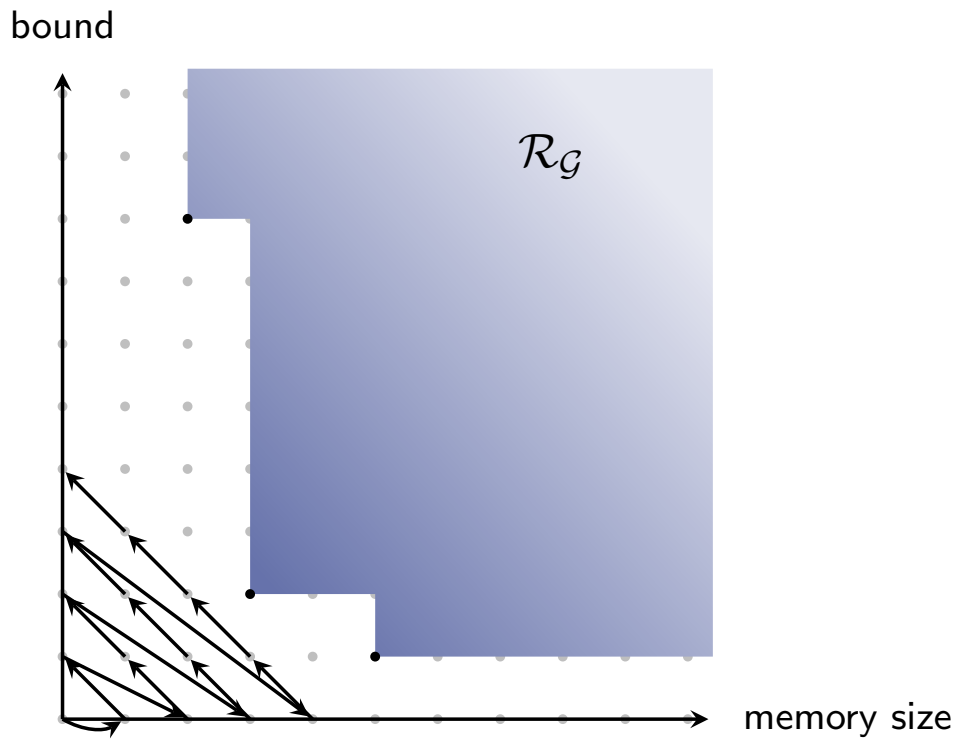
Tradeoffs



Implementation via bounded synthesis:

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

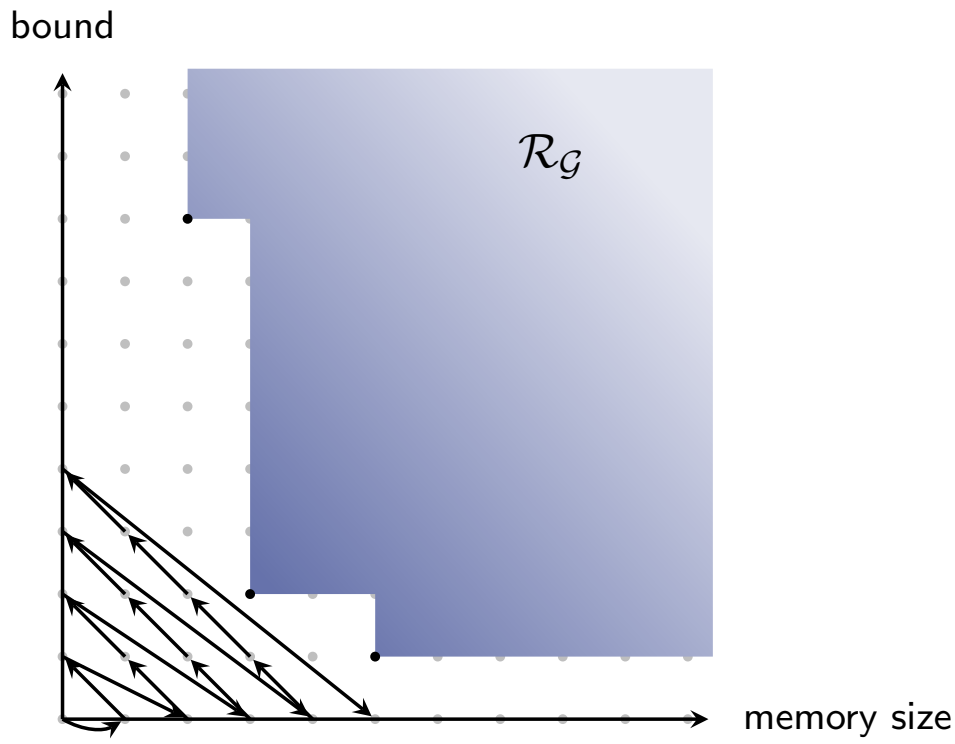
Tradeoffs



Implementation via bounded synthesis:

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

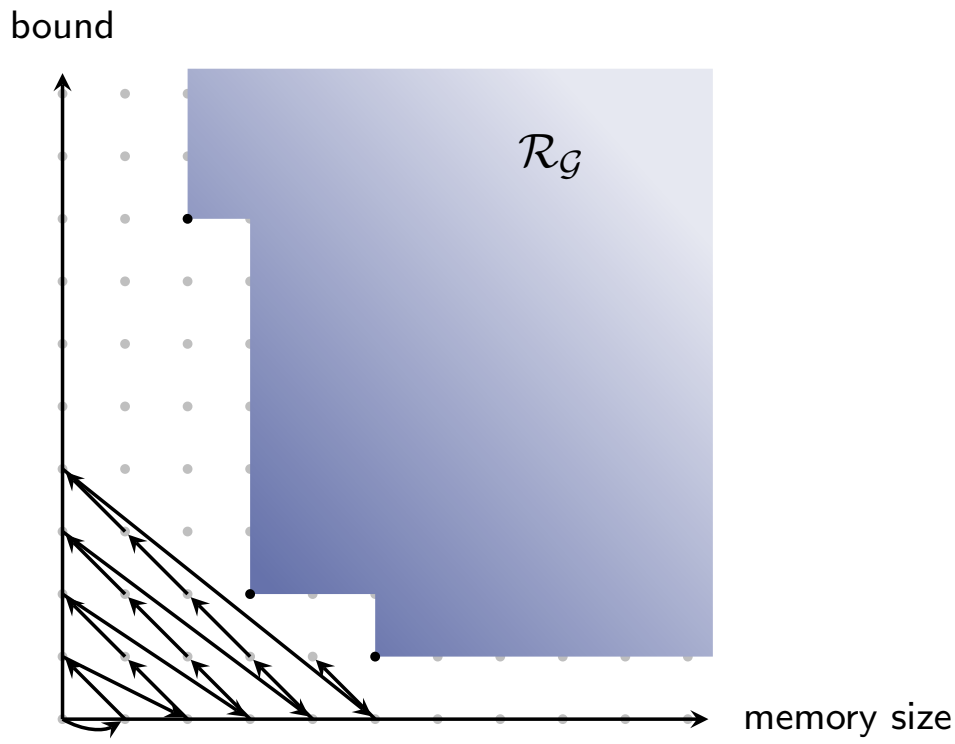
Tradeoffs



Implementation via bounded synthesis:

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

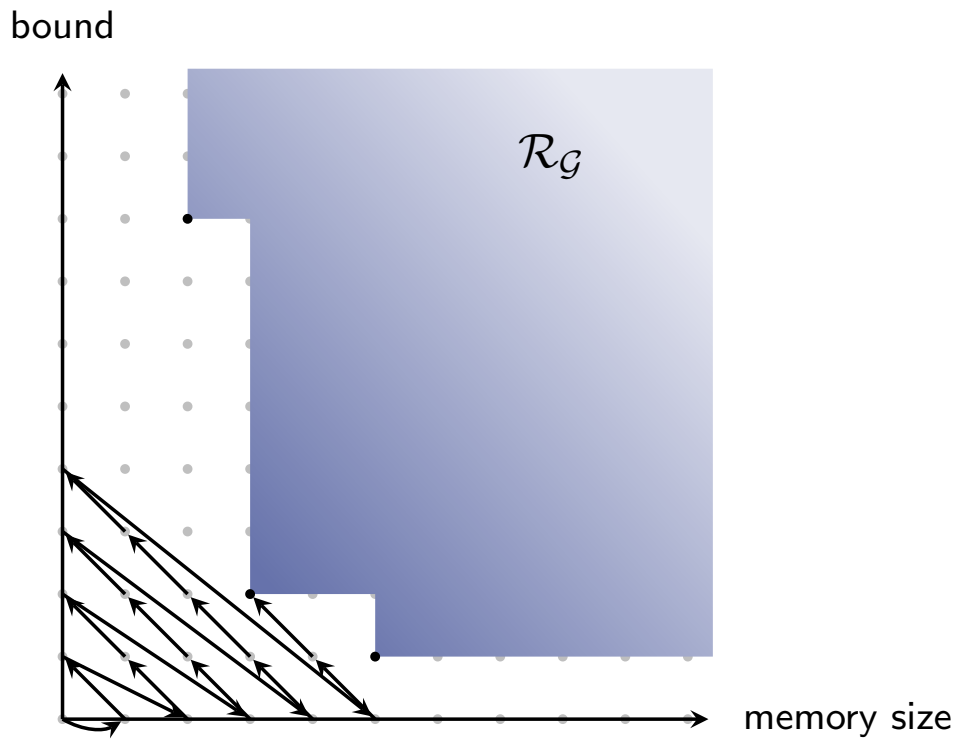
Tradeoffs



Implementation via bounded synthesis:

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

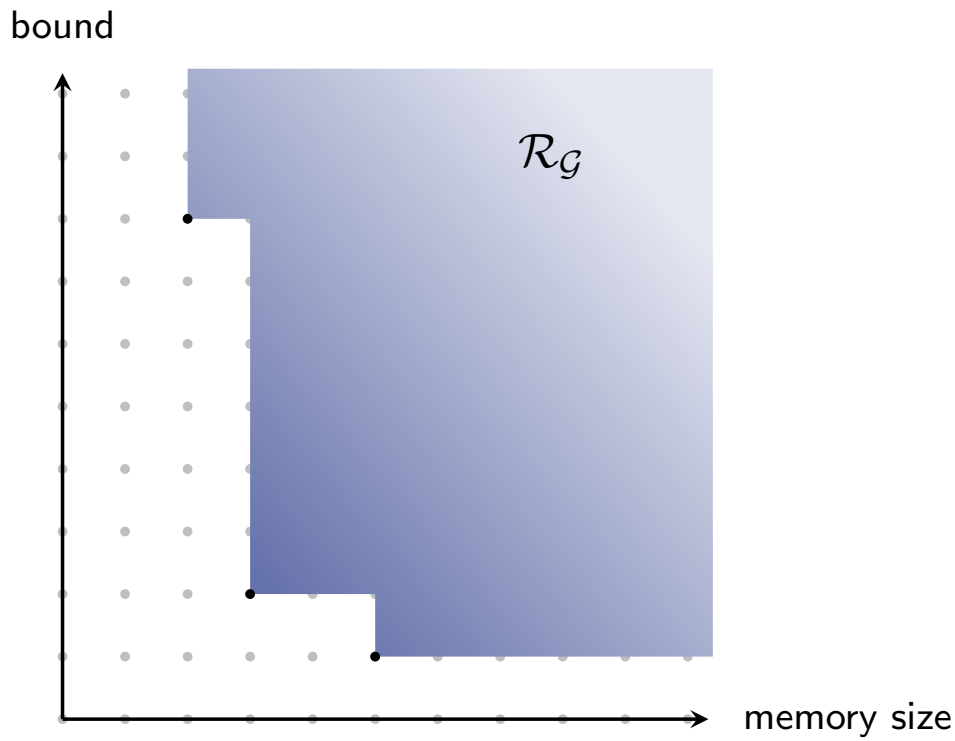
Tradeoffs



Implementation via bounded synthesis:

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

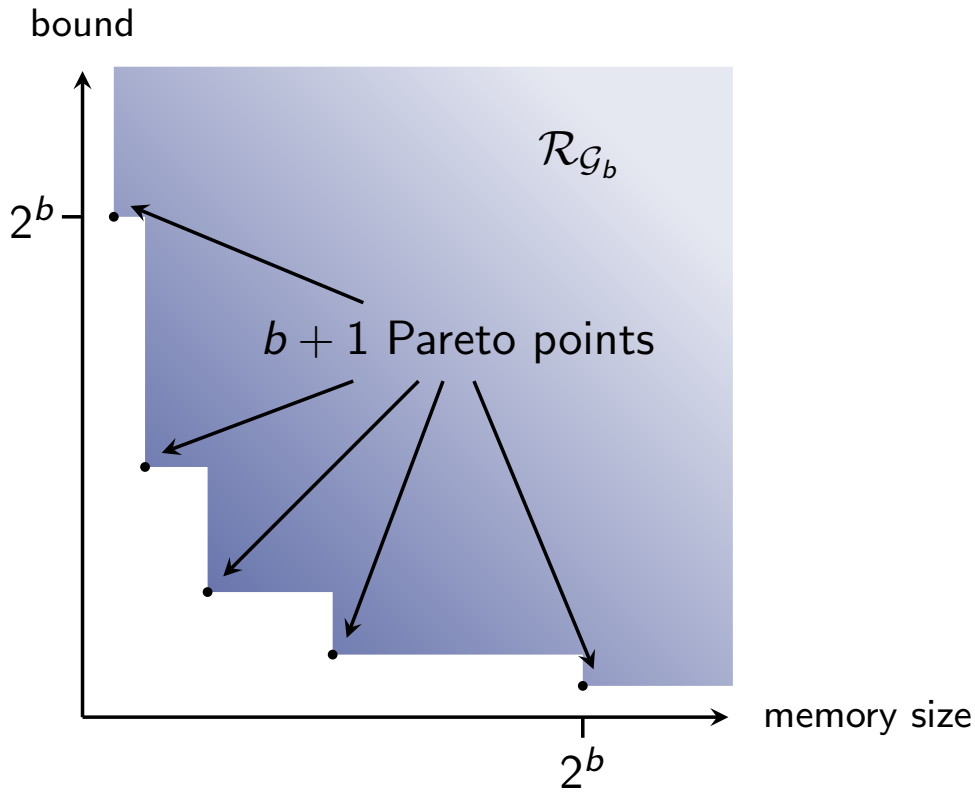
Tradeoffs



Lemma

1. *If* $(n, k) \in \mathcal{R}_\mathcal{G}$, *then* $(n, n2^{|\varphi|}) \in \mathcal{R}_\mathcal{G}$.
2. *If* $(n, k) \in \mathcal{R}_\mathcal{G}$, *then* $(2^{2^{k|\varphi|}}, k) \in \mathcal{R}_\mathcal{G}$.

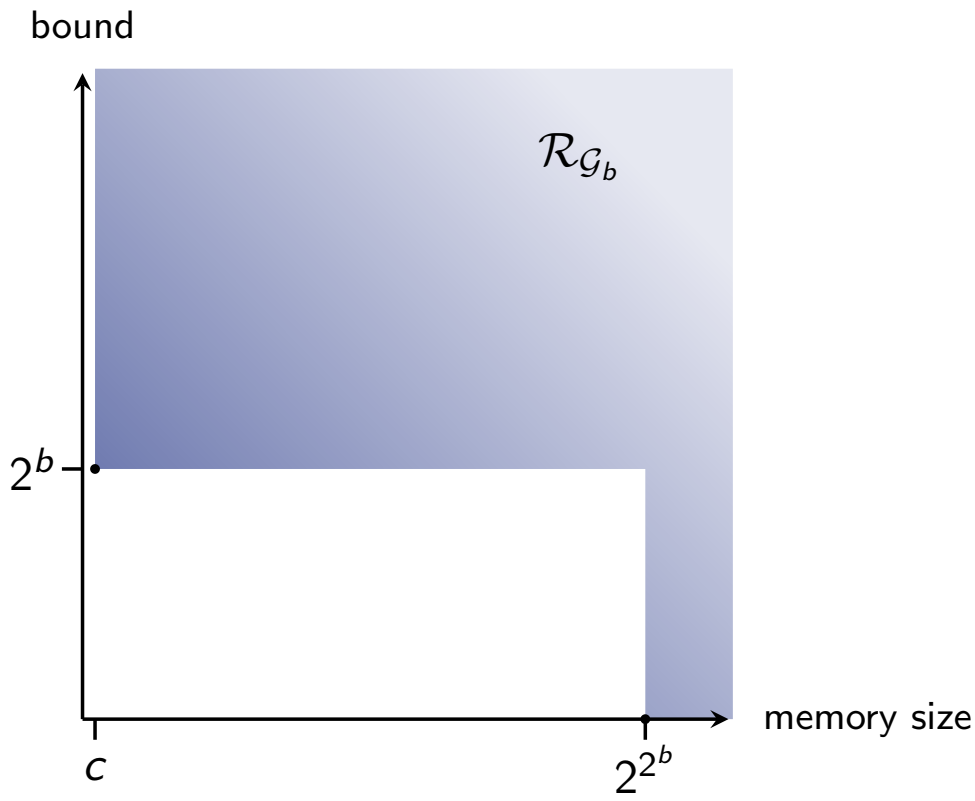
Tradeoffs



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 $j \leq b$.

Tradeoffs



Theorem (TWZ16)

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

An Example

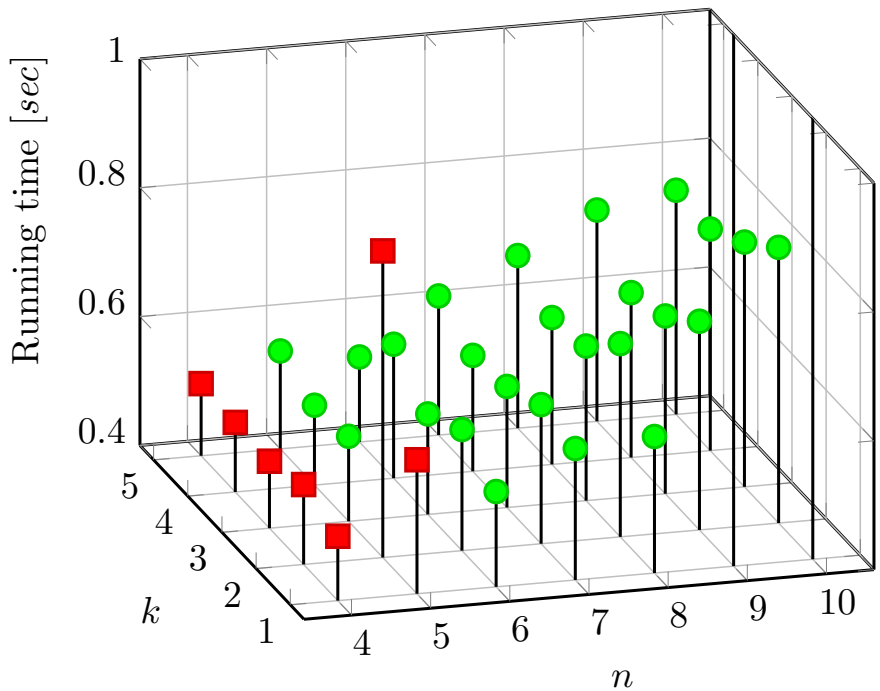
Five clients:

1. Answer every request of client 1 promptly: $\mathbf{G}\,(\mathtt{r}_1 \rightarrow \mathbf{F_P}\,\mathtt{g}_1)$
2. Answer every other request eventually: $\bigwedge_{i>1} \mathbf{G}\,(\mathtt{r}_i \rightarrow \mathbf{F}\,\mathtt{g}_i)$
3. At most one grant at a time: $\mathbf{G} \bigwedge_{i \neq j} \neg(\mathtt{g}_i \wedge \mathtt{g}_j)$

An Example

Five clients:

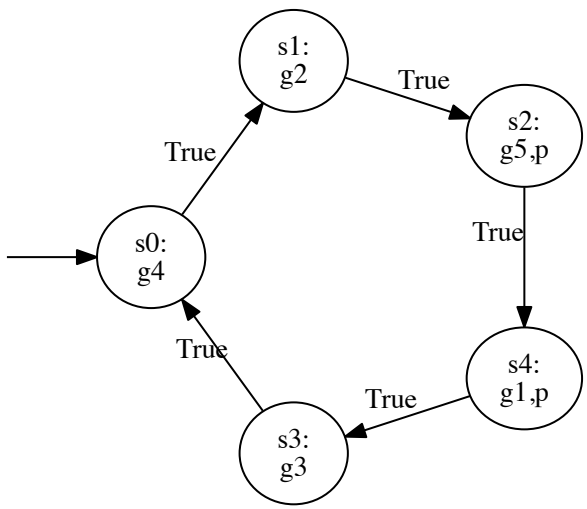
1. Answer every request of client 1 promptly: $\mathbf{G}\,(\mathtt{r}_1 \rightarrow \mathbf{F}_{\mathbf{P}}\,\mathtt{g}_1)$
2. Answer every other request eventually: $\bigwedge_{i>1} \mathbf{G}\,(\mathtt{r}_i \rightarrow \mathbf{F}\,\mathtt{g}_i)$
3. At most one grant at a time: $\mathbf{G} \bigwedge_{i \neq j} \neg(\mathtt{g}_i \wedge \mathtt{g}_j)$

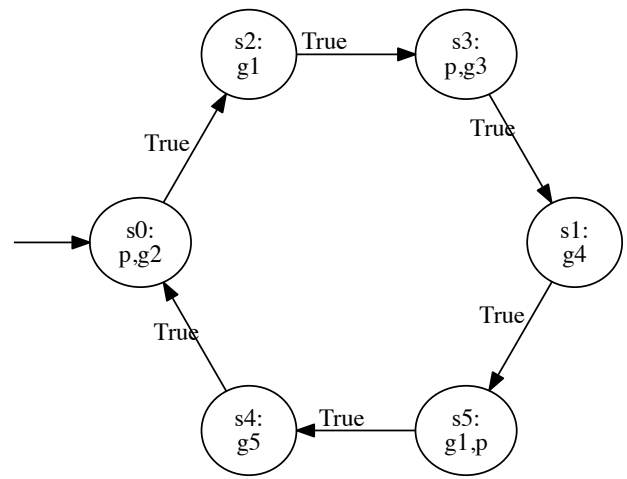


An Example

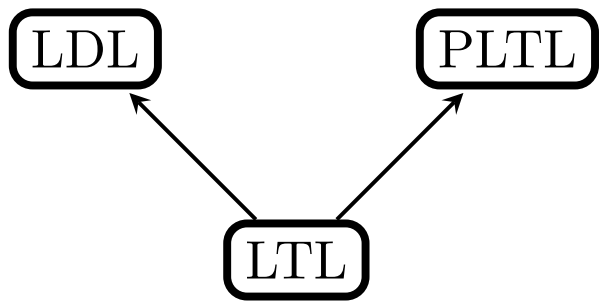
Five clients:

1. Answer every request of client 1 promptly: $\mathbf{G}\,(\mathtt{r}_1 \rightarrow \mathbf{F_P}\,\mathtt{g}_1)$
2. Answer every other request eventually: $\bigwedge_{i>1} \mathbf{G}\,(\mathtt{r}_i \rightarrow \mathbf{F}\,\mathtt{g}_i)$
3. At most one grant at a time: $\mathbf{G} \bigwedge_{i \neq j} \neg(\mathtt{g}_i \wedge \mathtt{g}_j)$



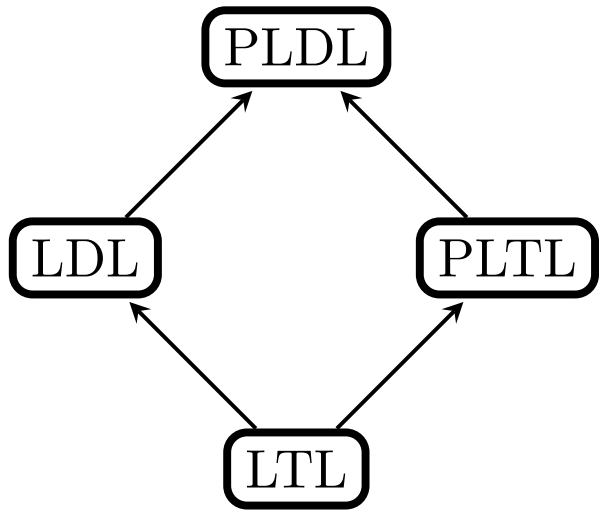


Generalizations



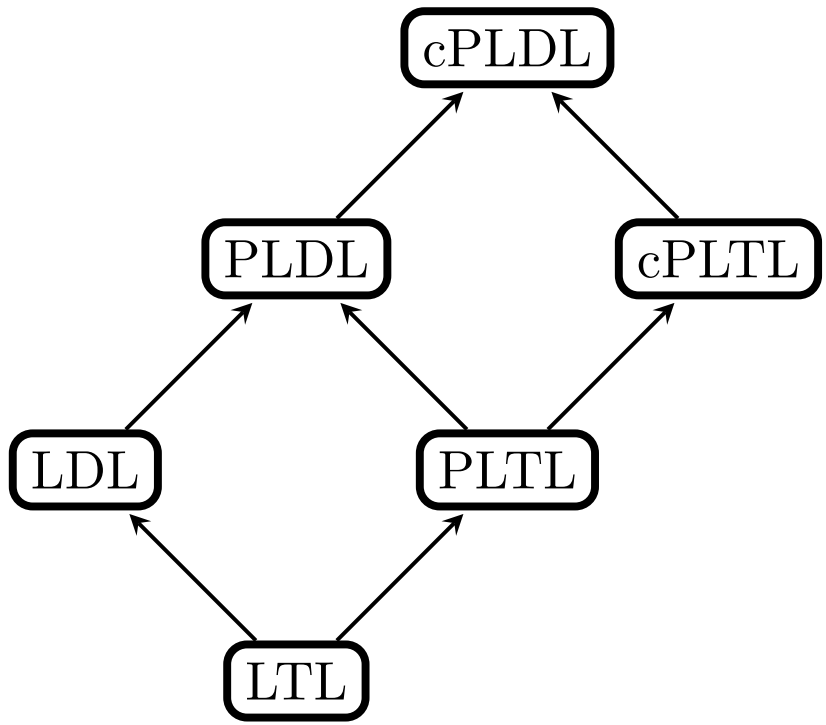
Generalizations

- Parametric LDL **[FZ14]**: full expressive power of the ω-regular languages and bounded operators.



Generalizations

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

Logic	Complexity	Memory Pl. 0 & 1
LTL	2ExpTime-complete	Doubly-exponential
LDL	2ExpTime-complete	Doubly-exponential
PLTL	2ExpTime-complete	Doubly-exponential
PLDL	2ExpTime-complete	Doubly-exponential
cPLTL	2ExpTime-complete	Doubly-exponential
cPLDL	2ExpTime-complete	Doubly-exponential

Generalizations

- Parametric LDL **[FZ14]**: full expressive power of the ω-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 ω-visibly pushdown languages.

Logic	Complexity	Memory Pl. 0 & 1
LTL	2ExpTime-complete	Doubly-exponential
LDL	2ExpTime-complete	Doubly-exponential
PLTL	2ExpTime-complete	Doubly-exponential
PLDL	2ExpTime-complete	Doubly-exponential
cPLTL	2ExpTime-complete	Doubly-exponential
cPLDL	2ExpTime-complete	Doubly-exponential
VLDL	3ExpTime-complete	pushdown transducer

Outline

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 ω-regular delay games is $\textsc{ExpTime}$*-complete and exponential lookahead is always sufficient and in general necessary.*

Best previous result: in $2\textsc{ExpTime}$ 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 $3\mathrm{ExpTime}$-complete, triply-exponential lookahead sufficient and necessary **[KZ16]**.
- Lookahead can be traded for quality in delay games with finitary parity conditions **[Z17]**.
- Finite-state strategies for delay games **[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**

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.

$\Rightarrow$ **Need to take tradeoffs into account when solving games.**