Approximating Optimal Bounds in Prompt-LTL Realizability in Doubly-exponential Time

Joint work with Leander Tentrup and Alexander Weinert

Martin Zimmermann

Saarland University

April, 2nd 2016
QAPL 16
Motivation

- Shift from programs to reactive systems:
  - non-terminating
  - interacting with a possibly antagonistic environment
  - communication-intensive
Motivation

- Shift from programs to reactive systems:
  - non-terminating
  - interacting with a possibly antagonistic environment
  - communication-intensive

- Successful approach to verification and synthesis: an infinite game between the system and its environment:
  - two players
  - infinite duration
  - perfect information
  - system player wins if specification is satisfied
Motivation

- Shift from programs to reactive systems:
  - non-terminating
  - interacting with a possibly antagonistic environment
  - communication-intensive

- Successful approach to verification and synthesis: an infinite game between the system and its environment:
  - two players
  - infinite duration
  - perfect information
  - system player wins if specification is satisfied

- Simplest model: realizability
Realizability: a Toy Example

- Setting: an arbiter with \( n \) clients
- requests \( r_i \) from client \( i \) controlled by the environment
- grants \( g_i \) for client \( i \) controlled by the system
Realizability: a Toy Example

- Setting: an arbiter with \( n \) clients
- requests \( r_i \) from client \( i \) controlled by the environment
- grants \( g_i \) for client \( i \) controlled by the system

Env:
Sys:
Realizability: a Toy Example

- Setting: an arbiter with $n$ clients
- requests $r_i$ from client $i$ controlled by the environment
- grants $g_i$ for client $i$ controlled by the system

Env: $r_1, r_2$
Sys:
Realizability: a Toy Example

- Setting: an arbiter with $n$ clients
- requests $r_i$ from client $i$ controlled by the environment
- grants $g_i$ for client $i$ controlled by the system

Env: $r_1, r_2$
Sys: $g_1$
Realizability: a Toy Example

- Setting: an arbiter with \( n \) clients
- requests \( r_i \) from client \( i \) controlled by the environment
- grants \( g_i \) for client \( i \) controlled by the system

Env: \( r_1, r_2, r_1 \)
Sys: \( g_1 \)
Realizability: a Toy Example

- Setting: an arbiter with $n$ clients
- requests $r_i$ from client $i$ controlled by the environment
- grants $g_i$ for client $i$ controlled by the system

```
Env:  r_1, r_2 \ r_1
Sys:   g_1 \ g_2
```
Realizability: a Toy Example

- Setting: an arbiter with \( n \) clients
- requests \( r_i \) from client \( i \) controlled by the environment
- grants \( g_i \) for client \( i \) controlled by the system

Env: \( r_1, r_2 \), \( r_1 \), \( r_1, r_4 \)
Sys: \( g_1 \), \( g_2 \)
Realizability: a Toy Example

- Setting: an arbiter with $n$ clients
- requests $r_i$ from client $i$ controlled by the environment
- grants $g_i$ for client $i$ controlled by the system

Environment (Env): $r_1, r_2, r_1, r_4$
System (Sys): $g_1, g_2, g_3$
Realizability: a Toy Example

- Setting: an arbiter with $n$ clients
- requests $r_i$ from client $i$ controlled by the environment
- grants $g_i$ for client $i$ controlled by the system

Env: $r_1, r_2$  $r_1$  $r_1, r_4$  $-$
Sys: $g_1$  $g_2$  $g_3$
Realizability: a Toy Example

- Setting: an arbiter with $n$ clients
- requests $r_i$ from client $i$ controlled by the environment
- grants $g_i$ for client $i$ controlled by the system

```
Env:   \( r_1, r_2 \quad r_1 \quad r_1, r_4 \quad \neg \)
Sys:   \( g_1 \quad g_2 \quad g_3 \quad g_4 \)
```
Realizability: a Toy Example

- Setting: an arbiter with \( n \) clients
- requests \( r_i \) from client \( i \) controlled by the environment
- grants \( g_i \) for client \( i \) controlled by the system

![Diagram of an arbiter with clients]

Env: \( r_1, r_2, r_1, r_4 \)  
Sys: \( g_1, g_2, g_3, g_4 \)
Realizability: a Toy Example

- Setting: an arbiter with $n$ clients
- requests $r_i$ from client $i$ controlled by the environment
- grants $g_i$ for client $i$ controlled by the system

Env: $r_1, r_2 \quad r_1 \quad r_1, r_4 \quad -$ \quad $-$

Sys: $g_1 \quad g_2 \quad g_3 \quad g_4 \quad g_1$
Realizability: a Toy Example

- Setting: an arbiter with $n$ clients
- requests $r_i$ from client $i$ controlled by the environment
- grants $g_i$ for client $i$ controlled by the system

Env: $r_1, r_2$  $r_1$  $r_1, r_4$  $-$  $-$  $-$
Sys: $g_1$  $g_2$  $g_3$  $g_4$  $g_1$
Realizability: a Toy Example

- Setting: an arbiter with \( n \) clients
- requests \( r_i \) from client \( i \) controlled by the environment
- grants \( g_i \) for client \( i \) controlled by the system

Env: \( r_1, r_2 \quad r_1 \quad r_1, r_4 \quad - \quad - \quad - \)

Sys: \( g_1 \quad g_2 \quad g_3 \quad g_4 \quad g_1 \quad g_2 \)
Realizability: a Toy Example

- Setting: an arbiter with \( n \) clients
- requests \( r_i \) from client \( i \) controlled by the environment
- grants \( g_i \) for client \( i \) controlled by the system

```
Env:  \( r_1, r_2 \)  \( r_1 \)  \( r_1, r_4 \)  \( - \)  \( - \)  \( - \)  \( - \)  \( r_2 \)
Sys:  \( g_1 \)  \( g_2 \)  \( g_3 \)  \( g_4 \)  \( g_1 \)  \( g_2 \)
```
Realizability: a Toy Example

- Setting: an arbiter with $n$ clients
- requests $r_i$ from client $i$ controlled by the environment
- grants $g_i$ for client $i$ controlled by the system

Env: $r_1, r_2$ $r_1$ $r_1, r_4$ $-$ $-$ $-$ $r_2$

Sys: $g_1$ $g_2$ $g_3$ $g_4$ $g_1$ $g_2$ $g_3$
Realizability: a Toy Example

- Setting: an arbiter with \( n \) clients
- requests \( r_i \) from client \( i \) controlled by the environment
- grants \( g_i \) for client \( i \) controlled by the system

Env: \( r_1, r_2 \) \( r_1 \) \( r_1, r_4 \) \( - \) \( - \) \( - \) \( r_2 \) \( r_1 \)
Sys: \( g_1 \) \( g_2 \) \( g_3 \) \( g_4 \) \( g_1 \) \( g_2 \) \( g_3 \)
Realizability: a Toy Example

- Setting: an arbiter with $n$ clients
- requests $r_i$ from client $i$ controlled by the environment
- grants $g_i$ for client $i$ controlled by the system

```
Env:  r_1, r_2  r_1  r_1, r_4  -  -  -  r_2  r_1
Sys:   g_1  g_2  g_3  g_4  g_1  g_2  g_3  g_4
```
Realizability: a Toy Example

- Setting: an arbiter with \( n \) clients
- requests \( r_i \) from client \( i \) controlled by the environment
- grants \( g_i \) for client \( i \) controlled by the system

![Diagram of the arbiter system with clients and requests]

**Env:** \( r_1, r_2, r_1, r_4, r_2, r_1 \)

**Sys:** \( g_1, g_2, g_3, g_4, g_1, g_2, g_3, g_4 \)
Realizability: a Toy Example

- Setting: an arbiter with \( n \) clients
- requests \( r_i \) from client \( i \) controlled by the environment
- grants \( g_i \) for client \( i \) controlled by the system

Env: \( r_1, r_2, r_1, r_4, \ldots, r_2, r_1, \ldots \)
Sys: \( g_1, g_2, g_3, g_4, g_1, g_2, g_3, g_4, g_2 \)
Realizability: a Toy Example

- Setting: an arbiter with $n$ clients
- requests $r_i$ from client $i$ controlled by the environment
- grants $g_i$ for client $i$ controlled by the system

Env: $r_1, r_2, r_1, r_4$; $- - - - r_2, r_1, - -$
Sys: $g_1, g_2, g_3, g_4, g_1, g_2, g_3, g_4, g_2$
Realizability: a Toy Example

- Setting: an arbiter with $n$ clients
- requests $r_i$ from client $i$ controlled by the environment
- grants $g_i$ for client $i$ controlled by the system

Env: $r_1, r_2, r_1, r_4, r_2, r_1$
Sys: $g_1, g_2, g_3, g_4, g_1, g_2, g_3, g_4, g_2, g_1$
Realizability: a Toy Example

- Setting: an arbiter with $n$ clients
- requests $r_i$ from client $i$ controlled by the environment
- grants $g_i$ for client $i$ controlled by the system

Env: $r_1, r_2$  $r_1$  $r_1, r_4$  $- - - - r_2$  $r_1$  $- - - -$

Sys: $g_1$  $g_2$  $g_3$  $g_4$  $g_1$  $g_2$  $g_3$  $g_4$  $g_2$  $g_1$
Realizability: a Toy Example

- Setting: an arbiter with \( n \) clients
- requests \( r_i \) from client \( i \) controlled by the environment
- grants \( g_i \) for client \( i \) controlled by the system

![Arbiter Diagram]

Env: \( r_1, r_2 \quad r_1 \quad r_1, r_4 \quad - \quad - \quad - \quad r_2 \quad r_1 \quad - \quad - \quad - \\
Sys: \quad g_1 \quad g_2 \quad g_3 \quad g_4 \quad g_1 \quad g_2 \quad g_3 \quad g_4 \quad g_2 \quad g_1 \quad -
Linear Temporal Logic

\[ \varphi ::= p \mid \neg p \mid \varphi \land \varphi \mid \varphi \lor \varphi \mid X \varphi \mid \varphi U \varphi \mid \varphi R \varphi \]

where \( p \) ranges over a finite set \( P \) of atomic propositions.
Linear Temporal Logic

\[ \varphi ::= p \mid \neg p \mid \varphi \land \varphi \mid \varphi \lor \varphi \mid X \varphi \mid \varphi \mathbf{U} \varphi \mid \varphi \mathbf{R} \varphi \]

where \( p \) ranges over a finite set \( P \) of atomic propositions.

Semantics: \( \rho \in (2^P)^\omega, \ n \in \mathbb{N} \)

- \((\rho, n) \models X \varphi : \rho \vdash \varphi \quad n \quad n+1 \)
- \((\rho, n) \models \psi \mathbf{U} \varphi : \rho \vdash \varphi \quad \psi \quad \psi \quad \psi \quad \varphi \quad \varphi \quad \varphi \quad \varphi \quad \varphi \quad \varphi \quad \cdots \)
- \((\rho, n) \models \psi \mathbf{R} \varphi : \rho \vdash \varphi \quad \varphi \quad \varphi \quad \varphi \quad \varphi \quad \varphi \quad \cdots \) \\
  \text{or} \\
  \rho \vdash \varphi \quad \varphi \quad \varphi \quad \varphi \quad \varphi, \psi
Use shorthands:

- $F \varphi = \text{tt} U \varphi$: eventually $\varphi$ holds
- $G \varphi = \text{ff} R \varphi$: $\varphi$ holds always
Continuing the Example: Specifications

Use shorthands:

- \( F \varphi = \texttt{tt} \ U \varphi \): eventually \( \varphi \) holds
- \( G \varphi = \texttt{ff} \ R \varphi \): \( \varphi \) holds always

Example specifications:

1. Answer every request: \( \bigwedge_i G (r_i \rightarrow F g_i) \)
Continuing the Example: Specifications

Use shorthands:

- \( \textbf{F} \varphi = \texttt{tt} \textbf{U} \varphi \): eventually \( \varphi \) holds
- \( \textbf{G} \varphi = \texttt{ff} \textbf{R} \varphi \): \( \varphi \) holds always

Example specifications:

1. Answer every request: \( \bigwedge_i \textbf{G} (r_i \rightarrow \textbf{F} g_i) \)
2. At most one grant at a time: \( \textbf{G} \bigwedge_{i \neq j} \neg (g_i \land g_j) \)
Continuing the Example: Specifications

Use shorthands:

- \( F \varphi = \mathsf{tt} \mathsf{U} \varphi \): eventually \( \varphi \) holds
- \( G \varphi = \mathsf{ff} \mathsf{R} \varphi \): \( \varphi \) holds always

Example specifications:

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:

\[
\bigwedge_i \neg [ (\neg r_i \mathsf{U} (\neg r_i \land g_i)) ] \land \neg [ F (g_i \land X (\neg r_i \mathsf{U} (\neg r_i \land g_i))) ]
\]
Continuing the Example: Specifications

Use shorthands:

- \( F \varphi = tt \ U \varphi \): eventually \( \varphi \) holds
- \( G \varphi = ff \ R \varphi \): \( \varphi \) holds always

Example specifications:

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:

\[
\bigwedge_i \neg \left[ \neg (r_i \ U (\neg r_i \land g_i)) \right] \land \neg \left[ F (g_i \land X (\neg r_i \ U (\neg r_i \land g_i))) \right]
\]

\[
\equiv \bigwedge_i \left[ (r_i \ R (r_i \lor \neg g_i)) \right] \land \left[ G (\neg g_i \lor X (r_i \ R (r_i \lor \neg g_i))) \right]
\]
Prompt-LTL

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

\[ \bigwedge_i G (r_i \rightarrow F g_i) \]
Problem: LTL is too weak to express timing-constraints: no guarantee when request is granted, only that it is granted eventually

\[ \bigwedge_i G (r_i \rightarrow F g_i) \]

Solution: add prompt-eventually operator \( F_P : \)

\[ \varphi ::= p \mid \neg p \mid \varphi \land \varphi \mid \varphi \lor \varphi \mid X \varphi \mid \varphi U \varphi \mid \varphi R \varphi \mid F_P \varphi \]
**Problem:** LTL is too weak to express timing-constraints: no guarantee when request is granted, only that it is granted eventually

\[ \bigwedge_i G (r_i \rightarrow F g_i) \]

**Solution:** 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 \( k \in \mathbb{N} \)

\[ (\rho, n, k) \models F_P \varphi : \rho \mid \cdots \mid n \varphi \mid n + k \]
Prompt-LTL

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

- \( \bigwedge_i G (r_i \rightarrow F g_i) \)

**Solution:** 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 \( k \in \mathbb{N} \)

- \( (\rho, n, k) \models F_P \varphi : \rho \ldots n \varphi n+k \)

Now: \( \bigwedge_i G (r_i \rightarrow F_P g_i) \)
Prompt-LTL Realizability

Given a Prompt-LTL formula $\varphi$, determine whether the system player has a strategy realizing $\varphi$ w.r.t. some bound $k$.

**Theorem (Kupferman et al. ’07)**

1. Prompt-LTL realizability is $2\text{EXPTIME}$-complete.
2. if $\varphi$ is realizable w.r.t. some $k$, then also w.r.t. $k_\varphi = 2^{2|\varphi|}$. 
Prompt-LTL Realizability

Given a Prompt-LTL formula $\varphi$, determine whether the system player has a strategy realizing $\varphi$ w.r.t. some bound $k$.

**Theorem (Kupferman et al. ’07)**

1. Prompt-LTL realizability is \(2\text{EXPTIME-complete}.\)
2. if $\varphi$ is realizable w.r.t. some $k$, then also w.r.t. $k_{\varphi} = 2^{2|\varphi|}$.

Prompt-LTL realizability as optimization problem: determine the smallest $k$ s.t. the system player has a strategy realizing $\varphi$ w.r.t. $k$.

**Theorem (Z. ’11)**

The Prompt-LTL realizability optimization problem can be solved in triply-exponential time.
The Alternating-color Technique

1. Add fresh proposition \( p \notin P \): think of a coloring.

2. Obtain \( \text{rel}(\varphi) \) by replacing each subformula \( \text{F}_P \psi \) of \( \varphi \) by

\[
(p \rightarrow (p \text{ U } (\neg p \text{ U } \text{rel}(\psi)))) \land (\neg p \rightarrow (\neg p \text{ U } (p \text{ U } \text{rel}(\psi))))
\]

Intuitively: \( \psi \) has to be satisfied within one color change.
The Alternating-color Technique

1. Add fresh proposition \( p \notin P \): think of a coloring.
2. Obtain \( \text{rel}(\varphi) \) by replacing each subformula \( F_P \psi \) of \( \varphi \) by

\[
(p \rightarrow (p \mathop{U}(\neg p \mathop{U} \text{rel}(\psi)))) \land (\neg p \rightarrow (\neg p \mathop{U}(p \mathop{U} \text{rel}(\psi)))).
\]

Intuitively: \( \psi \) has to be satisfied within one color change.
The Alternating-color Technique

1. Add fresh proposition \( p \notin P \): think of a coloring.
2. Obtain \( \text{rel}(\varphi) \) by replacing each subformula \( F_P \psi \) of \( \varphi \) by

\[
(p \rightarrow (p U (\neg p U \text{rel}(\psi)))) \land (\neg p \rightarrow (\neg p U (p U \text{rel}(\psi)))).
\]

Intuitively: \( \psi \) has to be satisfied within one color change.

\[
\text{rel}(F_P \psi) \leq k \quad \text{rel}(\psi) \leq k
\]

\[
F_P \psi \quad \psi \quad n \quad n + 2k
\]
The Alternating-color Technique

1. Add fresh proposition $p \notin P$: think of a coloring.
2. Obtain $\text{rel}(\varphi)$ by replacing each subformula $F_P \psi$ of $\varphi$ by

$$(p \rightarrow (p U (\neg p U \text{rel}(\psi)))) \land (\neg p \rightarrow (\neg p U (p U \text{rel}(\psi))))).$$

Intuitively: $\psi$ has to be satisfied within one color change.

Lemma (Kupferman et al. ’07)

Let $\varphi$ be a PROMPT–LTL formula, $w \in (2^P)^\omega$, and $w' \in (2^{P \cup \{p\}})^\omega$ s.t. $w$ and $w'$ coincide on $P$ at every position.

1. If $(w, k) \models \varphi$ and distance between color changes is at least $k$ in $w'$, then $w' \models \text{rel}(\varphi)$.

2. Let $k \in \mathbb{N}$. If $w' \models \text{rel}(\varphi)$ and distance between color-changes is at most $k$ in $w'$, then $(w, 2k) \models \varphi$. 
$$\psi_k$$ expressing that distance between color changes is at most $$k$$

**Lemma (Kupferman et al. ’07)**

*Let $$\varphi$$ be a PROMPT–LTL formula and let $$k \in \mathbb{N}$$.*

1. A strategy realizing $$\varphi$$ with respect to $$k$$ can be turned into a strategy realizing $$\text{rel}(\varphi) \land \psi_k$$.

2. A strategy realizing $$\text{rel}(\varphi) \land \psi_k$$ can be turned into a strategy realizing $$\varphi$$ with respect to $$2k$$.
Applying the Alternating-color Technique

\( \psi_k \) expressing that distance between color changes is at most \( k \)

**Lemma (Kupferman et al. '07)**

Let \( \varphi \) be a PROMPT–LTL formula and let \( k \in \mathbb{N} \).

1. A strategy realizing \( \varphi \) with respect to \( k \) can be turned into a strategy realizing \( \text{rel}(\varphi) \land \psi_k \).

2. A strategy realizing \( \text{rel}(\varphi) \land \psi_k \) can be turned into a strategy realizing \( \varphi \) with respect to \( 2k \).

**Lemma**

The following problem is in \( 2\text{Exptime} \): Given a PROMPT–LTL formula \( \varphi \) and a natural number \( k \leq 2^{2|\varphi|} \), is \( \text{rel}(\varphi) \land \psi_k \) realizable?
The Algorithm

1: if $\varphi$ unrealizable then
2: return “$\varphi$ unrealizable”
3: for $k = 0; k \leq 2^{2|\varphi|}; k \leftarrow k + 1$ do
4: if $\text{rel}(\varphi) \land \psi_k$ realizable then
5: return $2k$
The Algorithm

1: if $\varphi$ unrealizable then
2: return “$\varphi$ unrealizable”
3: for $k = 0; k \leq 2^{2|\varphi|}; k \leftarrow k + 1$ do
4: if $\text{rel}(\varphi) \land \psi_k$ realizable then
5: return $2k$

Run-time: doubly-exponential in $|\varphi|$: 
1. Lines 1 and 4: doubly-exponential time.
2. At most doubly-exponentially many iterations.
The Algorithm

1: if $\varphi$ unrealizable then
2: \hspace{1em} return “$\varphi$ unrealizable”
3: for $k = 0; k \leq 2^{2|\varphi|}; k \leftarrow k + 1$ do
4: \hspace{1em} if $\text{rel}(\varphi) \land \psi_k$ realizable then
5: \hspace{2em} return $2k$

Run-time: doubly-exponential in $|\varphi|$:
1. Lines 1 and 4: doubly-exponential time.
2. At most doubly-exponentially many iterations.

Approximation ratio:
\[
\frac{2k}{2k - k_{\text{opt}}} \leq \frac{2k}{2k - k} = 2.
\]
The Results

**Theorem**

*The optimization problem for PROMPT–LTL realizability can be approximated within a factor of 2 in doubly-exponential time. As a byproduct, one obtains a strategy witnessing the approximatively optimal bound.*
The Results

**Theorem**

The optimization problem for PROMPT–LTL realizability can be approximated within a factor of 2 in doubly-exponential time. As a byproduct, one obtains a strategy witnessing the approximatively optimal bound.

The same algorithm works for stronger logics as well

- **Parametric LTL**: allow multiple bounds on prompt-eventually: $F \leq x$ with parameter $x$ or on the dual operator $G \leq x$
- **Parametric LDL**: replace $F \leq x$ and $G \leq x$ by $\langle r \rangle \leq x$ and $[r] \leq x$ with regular expression $r$
Back to the Example

An arbiter with 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: \( G \bigwedge_{i \neq j} \neg (g_i \land g_j) \)
Bounded synthesis: incrementally search for smallest strategy

Two parameters: bound $k$ and size $n$ of strategy $\Rightarrow$ Tradeoffs
A Prototype Implementation

- Bounded synthesis: incrementally search for smallest strategy
- Two parameters: bound $k$ and size $n$ of strategy $\Rightarrow$ Tradeoffs
A Prototype Implementation

- Bounded synthesis: incrementally search for smallest strategy
- Two parameters: bound $k$ and size $n$ of strategy $\Rightarrow$ Tradeoffs
The Resulting Strategies

- \( k = 3 \Rightarrow \text{bound} \leq 6 \) and size \( n = 5 \)
- Implements round-robin strategy

- \( k = 1 \Rightarrow \text{bound} \leq 2 \) and size \( n = 6 \)
- Prioritizes client 1, others round-robin
Conclusion

Our contribution:

- The first approximation algorithm for Prompt-LTL realizability with doubly-exponential running time
- Computes a realizing strategy
- Applicable to stronger logics as well
- Not presented: tight exponential upper bounds on the tradeoff between bound and memory
- Preprint available at arXiv.
Conclusion

Our contribution:

- The first approximation algorithm for Prompt-LTL realizability with doubly-exponential running time
- Computes a realizing strategy
- Applicable to stronger logics as well
- Not presented: tight exponential upper bounds on the tradeoff between bound and memory
- Preprint available at arXiv.

Future work:

- Detailed experiments
- Study the tradeoffs between bound, size, and run time
- Show that the exact optimum can be computed in doubly-exponential time