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

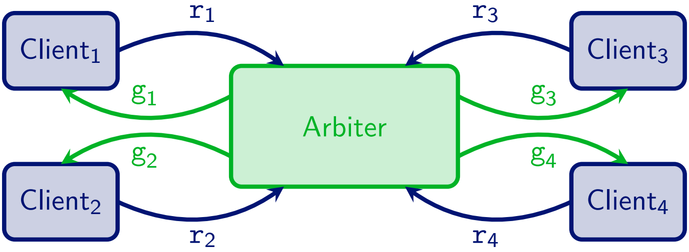
- Shift from programs to reactive systems:
  - non-terminating
  - interacting with a possibly antagonistic environment
  - communication-intensive

- 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

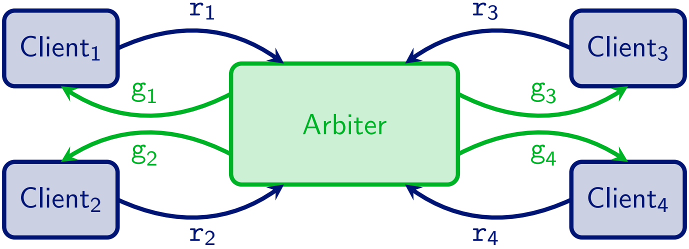
Realizability: a Toy Example

- Setting: an arbiter with n clients
- requests $\mathtt{r}_i$ from client i controlled by the environment
- grants $\mathtt{g}_i$ for client i controlled by the system



Realizability: a Toy Example

- Setting: an arbiter with n clients
- requests $\mathtt{r}_i$ from client i controlled by the environment
- grants $\mathtt{g}_i$ for client i controlled by the system

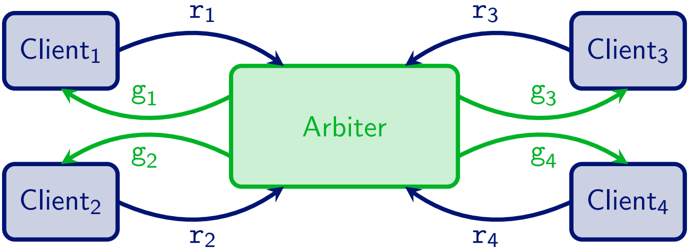


Env:

Sys:

Realizability: a Toy Example

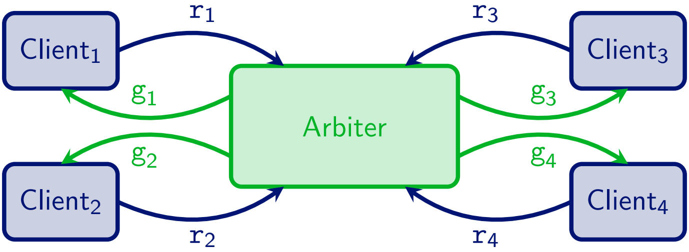
- Setting: an arbiter with n clients
- requests $\mathtt{r}_i$ from client i controlled by the environment
- grants $\mathtt{g}_i$ for client i controlled by the system



Env: $\mathtt{r}_1, \mathtt{r}_2$
Sys:

Realizability: a Toy Example

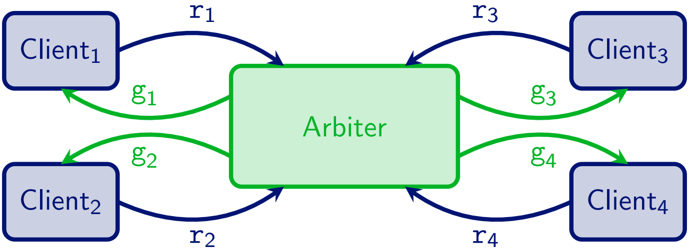
- Setting: an arbiter with n clients
- requests $\mathtt{r}_i$ from client i controlled by the environment
- grants $\mathtt{g}_i$ for client i controlled by the system



Env: $\mathtt{r}_1, \mathtt{r}_2$
Sys: $\mathtt{g}_1$

Realizability: a Toy Example

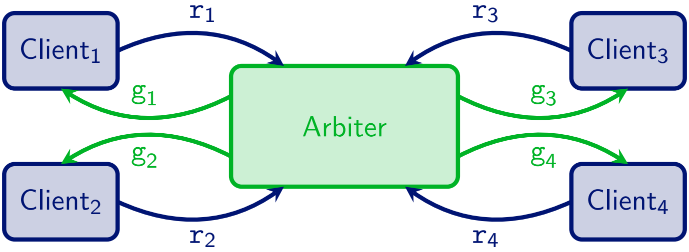
- Setting: an arbiter with n clients
- requests $\mathtt{r}_i$ from client i controlled by the environment
- grants $\mathtt{g}_i$ for client i controlled by the system



Env: $\mathtt{r}_1, \mathtt{r}_2$ $\mathtt{r}_1$
Sys: $\mathtt{g}_1$

Realizability: a Toy Example

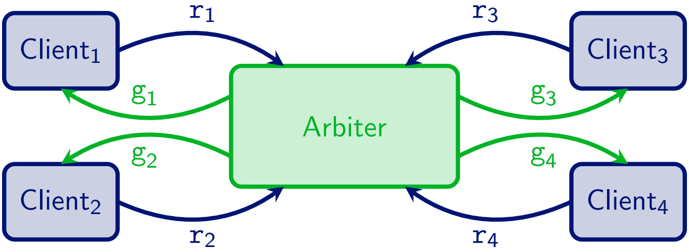
- Setting: an arbiter with n clients
- requests $\mathtt{r}_i$ from client i controlled by the environment
- grants $\mathtt{g}_i$ for client i controlled by the system



Env:	$\mathtt{r}_1, \mathtt{r}_2$	$\mathtt{r}_1$
Sys:	$\mathtt{g}_1$	$\mathtt{g}_2$

Realizability: a Toy Example

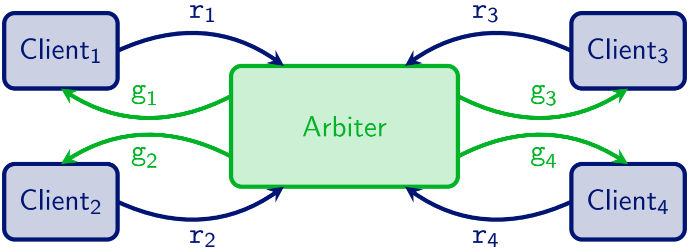
- Setting: an arbiter with n clients
- requests $\mathtt{r}_i$ from client i controlled by the environment
- grants $\mathtt{g}_i$ for client i controlled by the system



Env:	$\mathtt{r}_1, \mathtt{r}_2$	$\mathtt{r}_1$	$\mathtt{r}_1, \mathtt{r}_4$
Sys:	$\mathtt{g}_1$	$\mathtt{g}_2$	

Realizability: a Toy Example

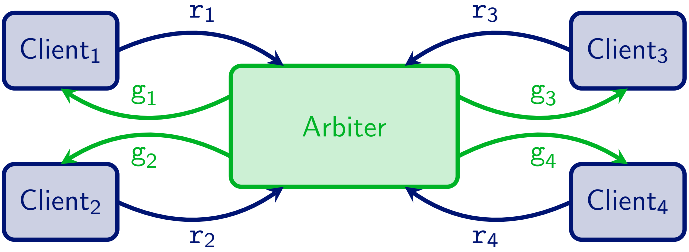
- Setting: an arbiter with n clients
- requests $\mathtt{r}_i$ from client i controlled by the environment
- grants $\mathtt{g}_i$ for client i controlled by the system



Env:	$\mathtt{r}_1, \mathtt{r}_2$	$\mathtt{r}_1$	$\mathtt{r}_1, \mathtt{r}_4$
Sys:	$\mathtt{g}_1$	$\mathtt{g}_2$	$\mathtt{g}_3$

Realizability: a Toy Example

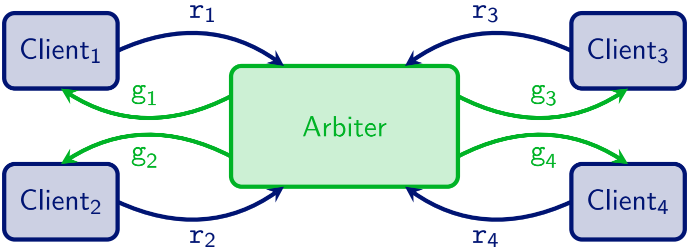
- Setting: an arbiter with n clients
- requests $\mathtt{r}_i$ from client i controlled by the environment
- grants $\mathtt{g}_i$ for client i controlled by the system



Env:	$\mathtt{r}_1, \mathtt{r}_2$	$\mathtt{r}_1$	$\mathtt{r}_1, \mathtt{r}_4$	–
Sys:	$\mathtt{g}_1$	$\mathtt{g}_2$	$\mathtt{g}_3$	

Realizability: a Toy Example

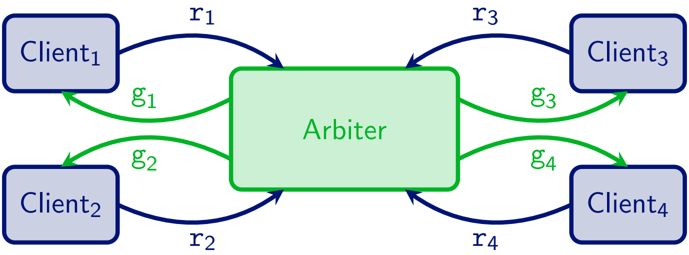
- Setting: an arbiter with n clients
- requests $\mathtt{r}_i$ from client i controlled by the environment
- grants $\mathtt{g}_i$ for client i controlled by the system



Env:	$\mathtt{r}_1, \mathtt{r}_2$	$\mathtt{r}_1$	$\mathtt{r}_1, \mathtt{r}_4$	—
Sys:	$\mathtt{g}_1$	$\mathtt{g}_2$	$\mathtt{g}_3$	$\mathtt{g}_4$

Realizability: a Toy Example

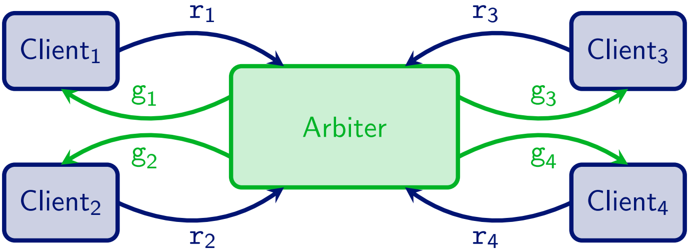
- Setting: an arbiter with n clients
- requests $\mathtt{r}_i$ from client i controlled by the environment
- grants $\mathtt{g}_i$ for client i controlled by the system



Env:	$\mathtt{r}_1, \mathtt{r}_2$	$\mathtt{r}_1$	$\mathtt{r}_1, \mathtt{r}_4$	—	—
Sys:	$\mathtt{g}_1$	$\mathtt{g}_2$	$\mathtt{g}_3$	$\mathtt{g}_4$	

Realizability: a Toy Example

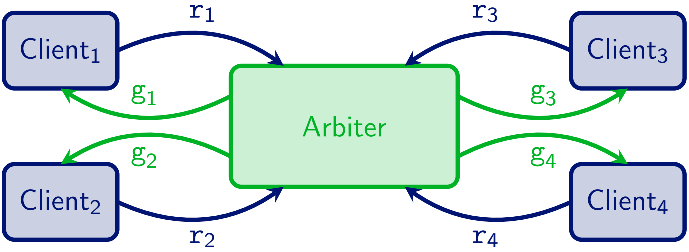
- Setting: an arbiter with n clients
- requests $\mathtt{r}_i$ from client i controlled by the environment
- grants $\mathtt{g}_i$ for client i controlled by the system



Env:	$\mathtt{r}_1, \mathtt{r}_2$	$\mathtt{r}_1$	$\mathtt{r}_1, \mathtt{r}_4$	—	—
Sys:	$\mathtt{g}_1$	$\mathtt{g}_2$	$\mathtt{g}_3$	$\mathtt{g}_4$	$\mathtt{g}_1$

Realizability: a Toy Example

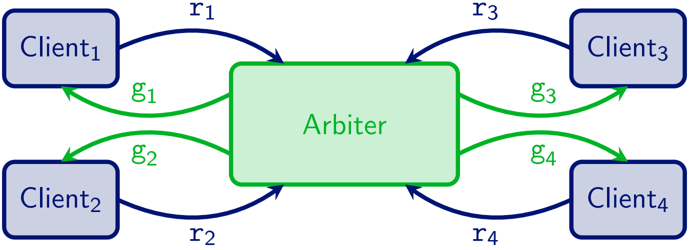
- Setting: an arbiter with n clients
- requests $\mathtt{r}_i$ from client i controlled by the environment
- grants $\mathtt{g}_i$ for client i controlled by the system



Env:	$\mathtt{r}_1, \mathtt{r}_2$	$\mathtt{r}_1$	$\mathtt{r}_1, \mathtt{r}_4$	—	—	—
Sys:	$\mathtt{g}_1$	$\mathtt{g}_2$	$\mathtt{g}_3$	$\mathtt{g}_4$	$\mathtt{g}_1$	

Realizability: a Toy Example

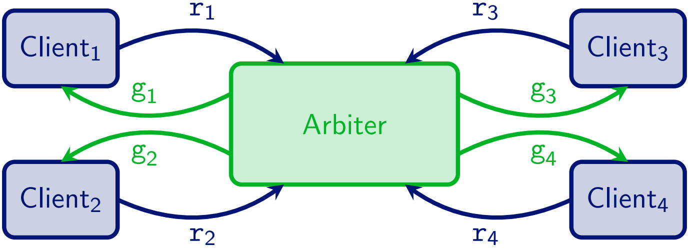
- Setting: an arbiter with n clients
- requests $\mathtt{r}_i$ from client i controlled by the environment
- grants $\mathtt{g}_i$ for client i controlled by the system



Env:	$\mathtt{r}_1, \mathtt{r}_2$	$\mathtt{r}_1$	$\mathtt{r}_1, \mathtt{r}_4$	—	—	—
Sys:	$\mathtt{g}_1$	$\mathtt{g}_2$	$\mathtt{g}_3$	$\mathtt{g}_4$	$\mathtt{g}_1$	$\mathtt{g}_2$

Realizability: a Toy Example

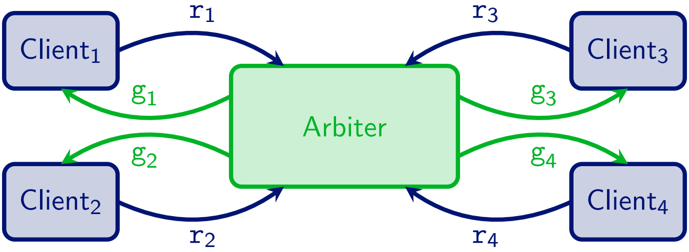
- Setting: an arbiter with n clients
- requests $\mathtt{r}_i$ from client i controlled by the environment
- grants $\mathtt{g}_i$ for client i controlled by the system



Env:	$\mathtt{r}_1, \mathtt{r}_2$	$\mathtt{r}_1$	$\mathtt{r}_1, \mathtt{r}_4$	—	—	—	$\mathtt{r}_2$
Sys:	$\mathtt{g}_1$	$\mathtt{g}_2$	$\mathtt{g}_3$	$\mathtt{g}_4$	$\mathtt{g}_1$	$\mathtt{g}_2$	

Realizability: a Toy Example

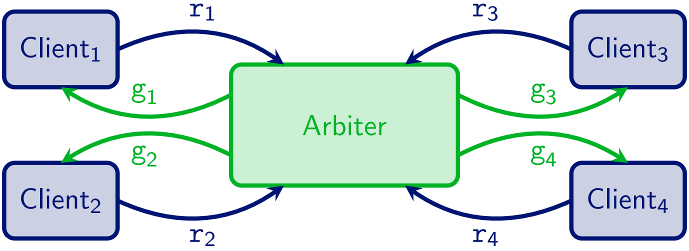
- Setting: an arbiter with n clients
- requests $\mathtt{r}_i$ from client i controlled by the environment
- grants $\mathtt{g}_i$ for client i controlled by the system



Env:	$\mathtt{r}_1, \mathtt{r}_2$	$\mathtt{r}_1$	$\mathtt{r}_1, \mathtt{r}_4$	—	—	—	$\mathtt{r}_2$
Sys:	$\mathtt{g}_1$	$\mathtt{g}_2$	$\mathtt{g}_3$	$\mathtt{g}_4$	$\mathtt{g}_1$	$\mathtt{g}_2$	$\mathtt{g}_3$

Realizability: a Toy Example

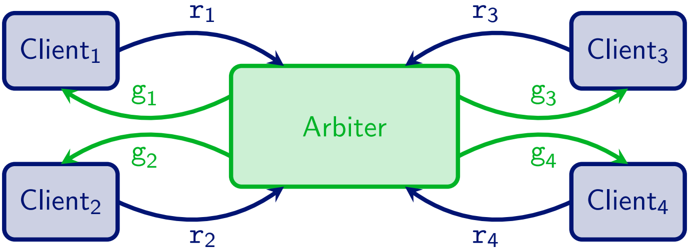
- Setting: an arbiter with n clients
- requests $\mathtt{r}_i$ from client i controlled by the environment
- grants $\mathtt{g}_i$ for client i controlled by the system



Env:	$\mathtt{r}_1, \mathtt{r}_2$	$\mathtt{r}_1$	$\mathtt{r}_1, \mathtt{r}_4$	—	—	—	$\mathtt{r}_2$	$\mathtt{r}_1$
Sys:	$\mathtt{g}_1$	$\mathtt{g}_2$	$\mathtt{g}_3$	$\mathtt{g}_4$	$\mathtt{g}_1$	$\mathtt{g}_2$	$\mathtt{g}_3$	

Realizability: a Toy Example

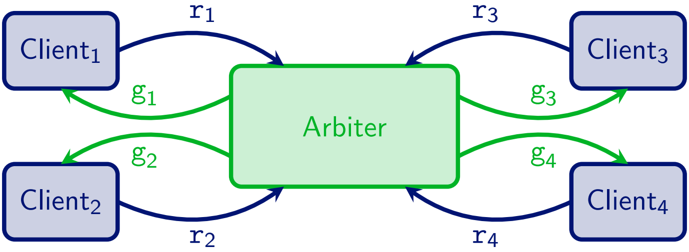
- Setting: an arbiter with n clients
- requests $\mathtt{r}_i$ from client i controlled by the environment
- grants $\mathtt{g}_i$ for client i controlled by the system



Env:	$\mathtt{r}_1, \mathtt{r}_2$	$\mathtt{r}_1$	$\mathtt{r}_1, \mathtt{r}_4$	—	—	—	$\mathtt{r}_2$	$\mathtt{r}_1$
Sys:	$\mathtt{g}_1$	$\mathtt{g}_2$	$\mathtt{g}_3$	$\mathtt{g}_4$	$\mathtt{g}_1$	$\mathtt{g}_2$	$\mathtt{g}_3$	$\mathtt{g}_4$

Realizability: a Toy Example

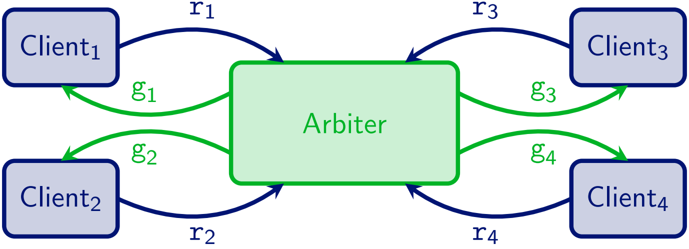
- Setting: an arbiter with n clients
- requests $\mathtt{r}_i$ from client i controlled by the environment
- grants $\mathtt{g}_i$ for client i controlled by the system



Env:	$\mathtt{r}_1, \mathtt{r}_2$	$\mathtt{r}_1$	$\mathtt{r}_1, \mathtt{r}_4$	—	—	—	$\mathtt{r}_2$	$\mathtt{r}_1$	—
Sys:	$\mathtt{g}_1$	$\mathtt{g}_2$	$\mathtt{g}_3$	$\mathtt{g}_4$	$\mathtt{g}_1$	$\mathtt{g}_2$	$\mathtt{g}_3$	$\mathtt{g}_4$	

Realizability: a Toy Example

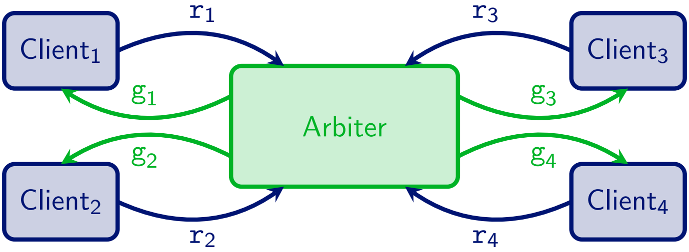
- Setting: an arbiter with n clients
- requests $\mathtt{r}_i$ from client i controlled by the environment
- grants $\mathtt{g}_i$ for client i controlled by the system



Env:	$\mathtt{r}_1, \mathtt{r}_2$	$\mathtt{r}_1$	$\mathtt{r}_1, \mathtt{r}_4$	—	—	—	$\mathtt{r}_2$	$\mathtt{r}_1$	—
Sys:	$\mathtt{g}_1$	$\mathtt{g}_2$	$\mathtt{g}_3$	$\mathtt{g}_4$	$\mathtt{g}_1$	$\mathtt{g}_2$	$\mathtt{g}_3$	$\mathtt{g}_4$	$\mathtt{g}_2$

Realizability: a Toy Example

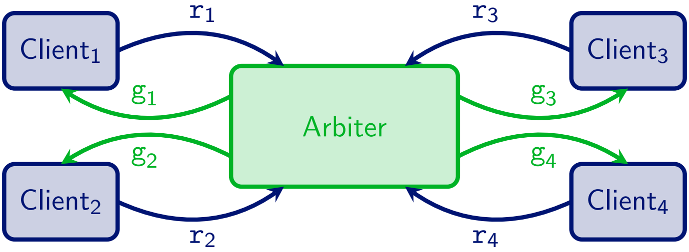
- Setting: an arbiter with n clients
- requests $\mathtt{r}_i$ from client i controlled by the environment
- grants $\mathtt{g}_i$ for client i controlled by the system



Env:	$\mathtt{r}_1, \mathtt{r}_2$	$\mathtt{r}_1$	$\mathtt{r}_1, \mathtt{r}_4$	—	—	—	$\mathtt{r}_2$	$\mathtt{r}_1$	—	—
Sys:	$\mathtt{g}_1$	$\mathtt{g}_2$	$\mathtt{g}_3$	$\mathtt{g}_4$	$\mathtt{g}_1$	$\mathtt{g}_2$	$\mathtt{g}_3$	$\mathtt{g}_4$	$\mathtt{g}_2$	

Realizability: a Toy Example

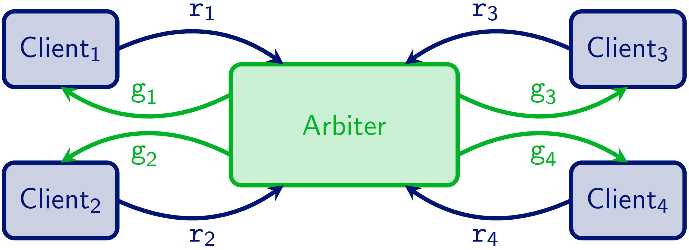
- Setting: an arbiter with n clients
- requests $\mathtt{r}_i$ from client i controlled by the environment
- grants $\mathtt{g}_i$ for client i controlled by the system



Env:	$\mathtt{r}_1, \mathtt{r}_2$	$\mathtt{r}_1$	$\mathtt{r}_1, \mathtt{r}_4$	—	—	—	$\mathtt{r}_2$	$\mathtt{r}_1$	—	—
Sys:	$\mathtt{g}_1$	$\mathtt{g}_2$	$\mathtt{g}_3$	$\mathtt{g}_4$	$\mathtt{g}_1$	$\mathtt{g}_2$	$\mathtt{g}_3$	$\mathtt{g}_4$	$\mathtt{g}_2$	$\mathtt{g}_1$

Realizability: a Toy Example

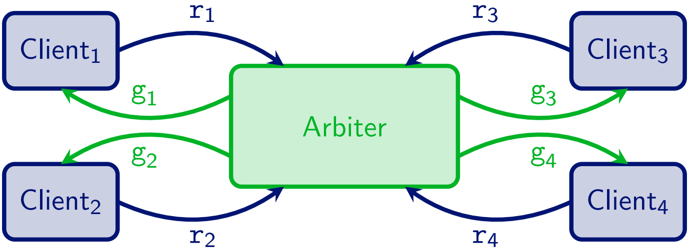
- Setting: an arbiter with n clients
- requests $\mathtt{r}_i$ from client i controlled by the environment
- grants $\mathtt{g}_i$ for client i controlled by the system



Env:	$\mathtt{r}_1, \mathtt{r}_2$	$\mathtt{r}_1$	$\mathtt{r}_1, \mathtt{r}_4$	—	—	—	$\mathtt{r}_2$	$\mathtt{r}_1$	—	—	—
Sys:	$\mathtt{g}_1$	$\mathtt{g}_2$	$\mathtt{g}_3$	$\mathtt{g}_4$	$\mathtt{g}_1$	$\mathtt{g}_2$	$\mathtt{g}_3$	$\mathtt{g}_4$	$\mathtt{g}_2$	$\mathtt{g}_1$	

Realizability: a Toy Example

- Setting: an arbiter with n clients
- requests $\texttt{r}_i$ from client i controlled by the environment
- grants $\texttt{g}_i$ for client i controlled by the system



Env:	$\texttt{r}_1, \texttt{r}_2$	$\texttt{r}_1$	$\texttt{r}_1, \texttt{r}_4$	—	—	—	$\texttt{r}_2$	$\texttt{r}_1$	—	—	—
Sys:	$\texttt{g}_1$	$\texttt{g}_2$	$\texttt{g}_3$	$\texttt{g}_4$	$\texttt{g}_1$	$\texttt{g}_2$	$\texttt{g}_3$	$\texttt{g}_4$	$\texttt{g}_2$	$\texttt{g}_1$	—

Linear Temporal Logic

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

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

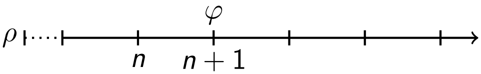
Linear Temporal Logic

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

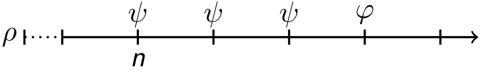
where p ranges over a finite set P of atomic propositions.

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

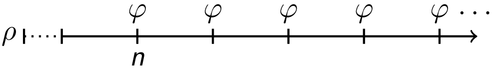
- $(\rho, n) \models \mathbf{X}\,\varphi$:



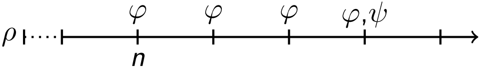
- $(\rho, n) \models \psi\,\mathbf{U}\,\varphi$:



- $(\rho, n) \models \psi\,\mathbf{R}\,\varphi$:



or



Continuing the Example: Specifications

Use shorthands:

- $\mathbf{F}\,\varphi = \mathbf{tt}\,\mathbf{U}\,\varphi$: eventually φ holds
- $\mathbf{G}\,\varphi = \mathbf{ff}\,\mathbf{R}\,\varphi$: φ holds always

Continuing the Example: Specifications

Use shorthands:

- $\mathbf{F}\,\varphi = \mathbf{tt}\,\mathbf{U}\,\varphi$: eventually φ holds
- $\mathbf{G}\,\varphi = \mathbf{ff}\,\mathbf{R}\,\varphi$: φ holds always

Example specifications:

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

Continuing the Example: Specifications

Use shorthands:

- $\mathbf{F}\,\varphi = \mathbf{tt}\,\mathbf{U}\,\varphi$: eventually φ holds
- $\mathbf{G}\,\varphi = \mathbf{ff}\,\mathbf{R}\,\varphi$: φ holds always

Example specifications:

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: $\mathbf{G}\,\bigwedge_{i \neq j} \neg(\mathtt{g}_i \wedge \mathtt{g}_j)$

Continuing the Example: Specifications

Use shorthands:

- $\mathbf{F}\,\varphi = \mathbf{tt}\,\mathbf{U}\,\varphi$: eventually φ holds
- $\mathbf{G}\,\varphi = \mathbf{ff}\,\mathbf{R}\,\varphi$: φ holds always

Example specifications:

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:

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

Continuing the Example: Specifications

Use shorthands:

- $\mathbf{F}\,\varphi = \mathbf{tt}\,\mathbf{U}\,\varphi$: eventually φ holds
- $\mathbf{G}\,\varphi = \mathbf{ff}\,\mathbf{R}\,\varphi$: φ holds always

Example specifications:

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:

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

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

Prompt-LTL

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

- $\bigwedge_i \mathbf{G}\,(\mathtt{r}_i \rightarrow \mathbf{F}\,\mathtt{g}_i)$

Prompt-LTL

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

- $\bigwedge_i \mathbf{G}\,(\mathtt{r}_i \rightarrow \mathbf{F}\,\mathtt{g}_i)$

Solution: 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

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

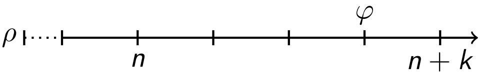
- $\bigwedge_i \mathbf{G}\,(\mathtt{r}_i \to \mathbf{F}\,\mathtt{g}_i)$

Solution: 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

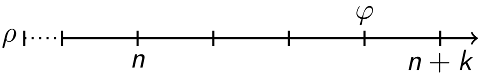
Problem: LTL is too weak to express timing-constraints: no guarantee when request is granted, only that it is granted eventually

- $\bigwedge_i \mathbf{G}\,(\mathtt{r}_i \rightarrow \mathbf{F}\,\mathtt{g}_i)$

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

Prompt-LTL Realizability

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

Theorem (Kupferman et al. '07)

1. *Prompt-LTL realizability is* $2\textsc{Exptime}$*-complete.*
2. *if* φ *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 φ, determine whether the system player has a strategy realizing φ w.r.t. some bound k.

Theorem (Kupferman et al. '07)

1. *Prompt-LTL realizability is* $2\textsc{Exptime}$*-complete.*
2. *if* φ *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 φ 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 $\mathrm{rel}(\varphi)$ by replacing each subformula $\mathbf{F_P}\,\psi$ of φ by

$$(p \to (p \,\mathbf{U}\, (\neg p \,\mathbf{U}\, \mathrm{rel}(\psi)))) \wedge (\neg p \to (\neg p \,\mathbf{U}\, (p \,\mathbf{U}\, \mathrm{rel}(\psi)))).$$

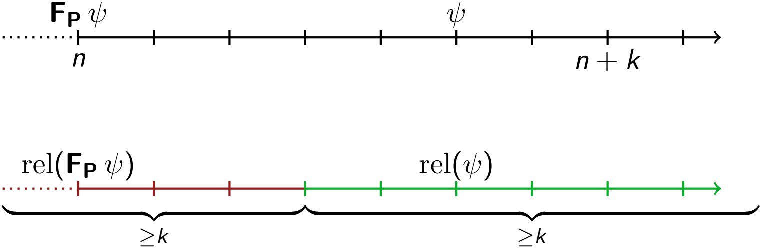
   Intuitively: ψ has to be satisfied within one color change.

The Alternating-color Technique

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

$$(p \rightarrow (p\,\mathbf{U}\,(\neg p\,\mathbf{U}\,\mathrm{rel}(\psi)))) \wedge (\neg p \rightarrow (\neg p\,\mathbf{U}\,(p\,\mathbf{U}\,\mathrm{rel}(\psi)))).$$

Intuitively: ψ has to be satisfied within one color change.

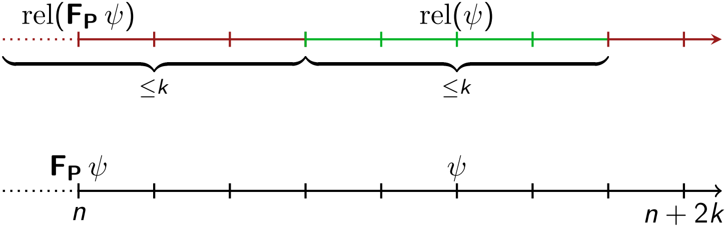


The Alternating-color Technique

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

$$(p \rightarrow (p\,\mathbf{U}\,(\neg p\,\mathbf{U}\,\mathrm{rel}(\psi)))) \wedge (\neg p \rightarrow (\neg p\,\mathbf{U}\,(p\,\mathbf{U}\,\mathrm{rel}(\psi)))).$$

Intuitively: ψ 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 $\mathrm{rel}(\varphi)$ by replacing each subformula $\mathbf{F_P}\,\psi$ of φ by

$$(p \rightarrow (p\,\mathbf{U}\,(\neg p\,\mathbf{U}\,\mathrm{rel}(\psi)))) \wedge (\neg p \rightarrow (\neg p\,\mathbf{U}\,(p\,\mathbf{U}\,\mathrm{rel}(\psi)))).$$

   Intuitively: ψ has to be satisfied within one color change.

Lemma (Kupferman et al. '07)

Let φ be a $\mathrm{PROMPT\text{--}LTL}$ 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 $w' \models \mathrm{rel}(\varphi)$.*
2. *Let $k \in \mathbb{N}$. If $w' \models \mathrm{rel}(\varphi)$ and distance between color-changes is at most k in w', then $(w, 2k) \models \varphi$.*

Applying the Alternating-color Technique

ψ_k expressing that distance between color changes is at most k

Lemma (Kupferman et al. '07)

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

1. *A strategy realizing* φ *with respect to* k *can be turned into a strategy realizing* $\mathrm{rel}(\varphi) \wedge \psi_k$.
2. *A strategy realizing* $\mathrm{rel}(\varphi) \wedge \psi_k$ *can be turned into a strategy realizing* φ *with respect to* $2k$.

Applying the Alternating-color Technique

ψ_k expressing that distance between color changes is at most k

Lemma (Kupferman et al. '07)

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

1. *A strategy realizing φ with respect to k can be turned into a strategy realizing $\mathrm{rel}(\varphi) \wedge \psi_k$.*
2. *A strategy realizing $\mathrm{rel}(\varphi) \wedge \psi_k$ can be turned into a strategy realizing φ with respect to $2k$.*

Lemma

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

The Algorithm

if φ unrealizable **then**
    return "φ unrealizable"
for $k = 0;\ k \leq 2^{2^{|\varphi|}};\ k \leftarrow k + 1$ **do**
    if $\mathrm{rel}(\varphi) \wedge \psi_k$ realizable **then**
        return $2k$

The Algorithm

1: **if** φ unrealizable **then**

2: **return** "φ unrealizable"

3: **for** $k = 0$; $k \leq 2^{2^{|\varphi|}}$; $k \leftarrow k + 1$ **do**

4: **if** $\mathrm{rel}(\varphi) \wedge \psi_k$ 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 φ unrealizable then
2:     return "φ unrealizable"
3: for k = 0; k ≤ 2^(2^|φ|); k ← k + 1 do
4:     if rel(φ) ∧ ψ_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.

Approximation ratio:

$$\frac{2k}{2k - k_{\mathrm{opt}}} \leq \frac{2k}{2k - k} = 2.$$

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 $\mathrm{PROMPT–LTL}$ *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: $\mathbf{F}_{\leq x}$ with parameter x or on the dual operator $\mathbf{G}_{\leq x}$
- **Parametric LDL:** replace $\mathbf{F}_{\leq x}$ and $\mathbf{G}_{\leq x}$ by $\langle r \rangle_{\leq x}$ and $[r]_{\leq x}$ with regular expression r

Back to the Example

An arbiter with five clients:

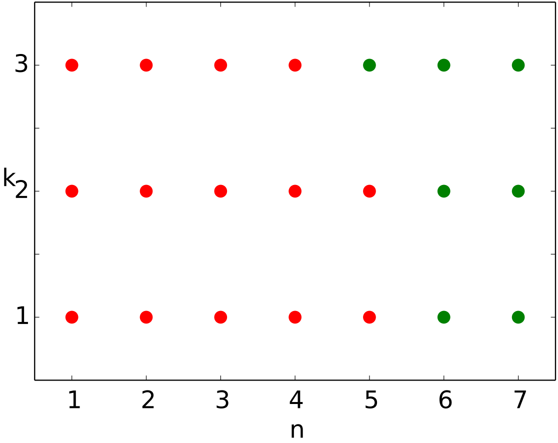
1. Answer every request of client 1 promptly: $\mathbf{G}\,(\mathtt{r}_1 \rightarrow \mathbf{F_P}\,\mathtt{g}_1)$
2. Answer every other request eventually: $\bigwedge_{i>1} \mathbf{G}\,(\mathtt{r}_i \rightarrow \mathbf{F}\,\mathtt{g}_i)$
3. At most one grant at a time: $\mathbf{G} \bigwedge_{i\neq j}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

A Prototype Implementation

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

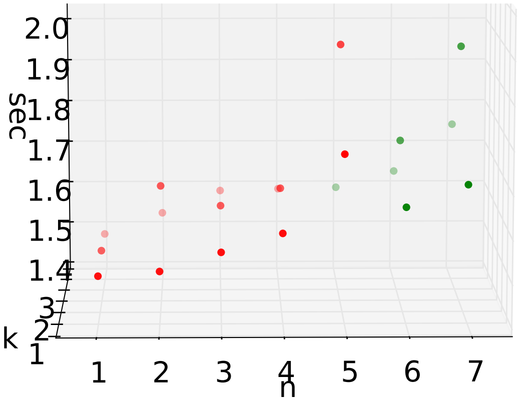
A Prototype Implementation

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



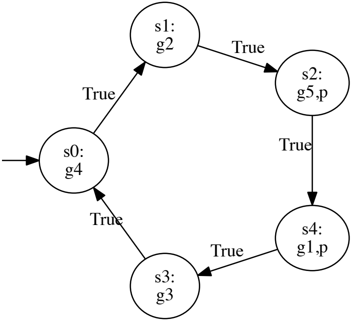
A Prototype Implementation

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

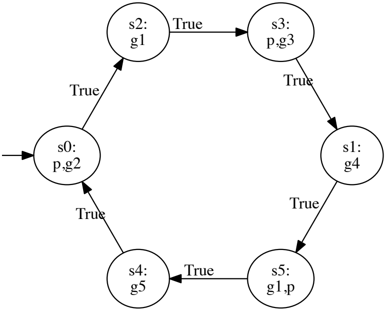


The Resulting Strategies

- $k = 3 \Rightarrow$ bound ≤ 6 and size $n = 5$
- Implements round-robin strategy



- $k = 1 \Rightarrow$ bound ≤ 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