Synthesis of succinct strategies thanks to ready simulation

Gilles Geeraerts, Joël Goossens, **Amélie Stainer**

Département d'Informatique, Université Libre de Bruxelles
Bruxelles, BELGIQUE

CASSTING, Grenoble – April 2014

Motivation

Safety games have a lot of applications:

- scheduler synthesis
- LTL synthesis
- determinization of timed automata

Motivation

Safety games have a lot of applications:

- scheduler synthesis
- LTL synthesis
- determinization of timed automata

Common points:

- a large arena implicitly defined
- a particular structure

Motivation

Safety games have a lot of applications:

- scheduler synthesis
- LTL synthesis
- determinization of timed automata

Common points:

- a large arena implicitly defined
- a particular structure

Goals:

- succinct winning strategies ($\sim$ implementability)
- effective construction

Motivation

Safety games have a lot of applications:

- scheduler synthesis
- LTL synthesis
- determinization of timed automata

Common points:

- a large arena implicitly defined
- a particular structure

Goals:

- succinct winning strategies ($\sim$ implementability)
- effective construction

$\Rightarrow$ using the structure

Outline

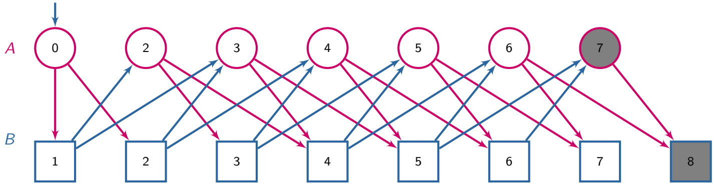
1. Introduction to safety games

2. Structured games

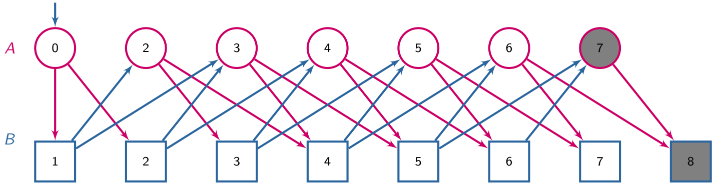
3. Succinct strategies in structured games

4. Efficient construction

Finite turn-based safety games

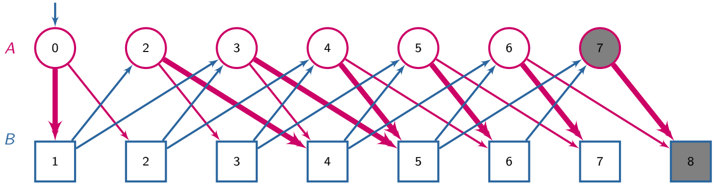


Finite turn-based safety games



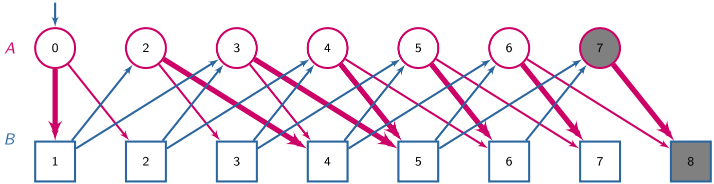
- Strategy of A: $\sigma : V_A \to V_B$

Finite turn-based safety games



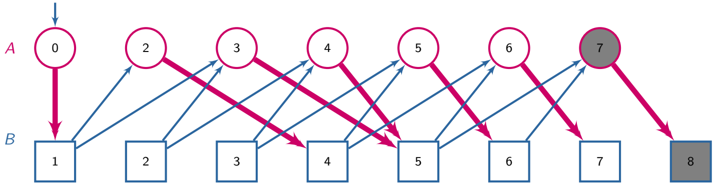
- Strategy of A: $\sigma : V_A \to V_B$

Finite turn-based safety games



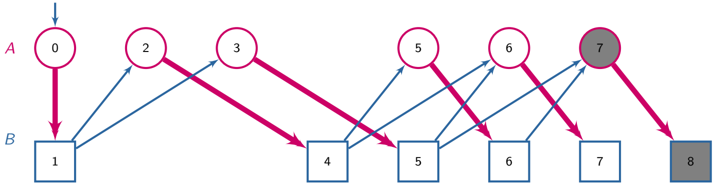
- Strategy of A: $\sigma : V_A \to V_B$
- G_σ: the arena with only moves of σ

Finite turn-based safety games



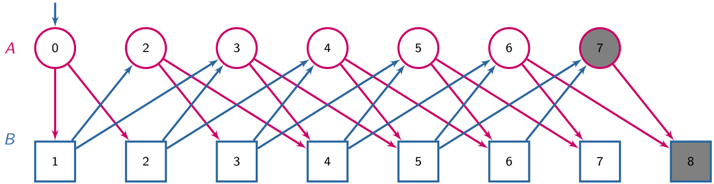
- Strategy of A: $\sigma : V_A \to V_B$
- G_σ: the arena with only moves of σ

Finite turn-based safety games



- Strategy of A: $\sigma : V_A \to V_B$
- G_σ: the arena with only moves of σ
- Winning strategy for A: Bad is not reachable in G_σ

Finite turn-based safety games

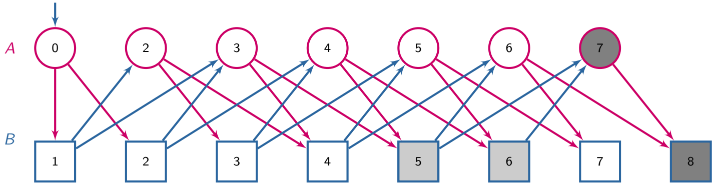


- Strategy of A: $\sigma : V_A \to V_B$
- G_σ: the arena with only moves of σ
- Winning strategy for A: Bad is not reachable in G_σ

Computation of winning and losing states

- Attractors of bad states:
  - $\mathrm{Attr}_0 = \mathrm{Bad}$
  - $\mathrm{Attr}_{i+1} = \mathrm{Attr}_i \cup \{v \in V_B | \exists v' \in \mathrm{Attr}_i, v \to v'\}$
    $\cup \{v \in V_A | \forall v' \in V_B \text{s.t.} v \to v', v' \in \mathrm{Attr}_i\}$
- $\mathrm{Lose} = (\cup_i \mathrm{Attr}_i)$; $\mathrm{Win} = \mathrm{Lose}^c$

Finite turn-based safety games

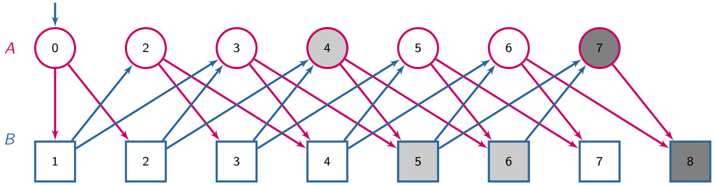


- Strategy of A: $\sigma : V_A \to V_B$
- G_σ: the arena with only moves of σ
- Winning strategy for A: Bad is not reachable in G_σ

Computation of winning and losing states

- Attractors of bad states:
  - $\mathrm{Attr}_0 = \mathrm{Bad}$
  - $\mathrm{Attr}_{i+1} = \mathrm{Attr}_i \cup \{v \in V_B | \exists v' \in \mathrm{Attr}_i, v \to v'\}$
    $\cup \{v \in V_A | \forall v' \in V_B \text{s.t.} v \to v', v' \in \mathrm{Attr}_i\}$
- $\mathrm{Lose} = (\cup_i \mathrm{Attr}_i)$; $\mathrm{Win} = \mathrm{Lose}^c$

Finite turn-based safety games

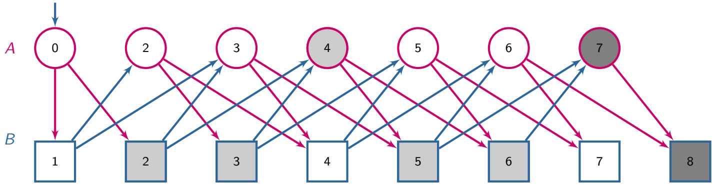


- Strategy of A: $\sigma : V_A \to V_B$
- G_σ: the arena with only moves of σ
- Winning strategy for A: Bad is not reachable in G_σ

Computation of winning and losing states

- Attractors of bad states:
  - $\mathrm{Attr}_0 = \mathrm{Bad}$
  - $\mathrm{Attr}_{i+1} = \mathrm{Attr}_i \cup \{v \in V_B | \exists v' \in \mathrm{Attr}_i, v \to v'\}$
    $\cup \{v \in V_A | \forall v' \in V_B \text{s.t.}\, v \to v', v' \in \mathrm{Attr}_i\}$
- $\mathrm{Lose} = (\cup_i \mathrm{Attr}_i)$; $\mathrm{Win} = \mathrm{Lose}^c$

Finite turn-based safety games

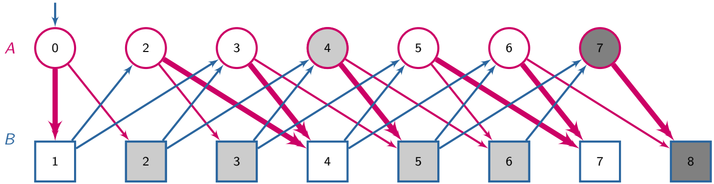


- Strategy of A: $\sigma : V_A \to V_B$
- G_σ: the arena with only moves of σ
- Winning strategy for A: Bad is not reachable in G_σ

Computation of winning and losing states

- Attractors of bad states:
  - $\mathrm{Attr}_0 = \mathrm{Bad}$
  - $\mathrm{Attr}_{i+1} = \mathrm{Attr}_i \cup \{v \in V_B | \exists v' \in \mathrm{Attr}_i, v \to v'\}$
    $\cup \{v \in V_A | \forall v' \in V_B \text{s.t.} v \to v', v' \in \mathrm{Attr}_i\}$
- $\mathrm{Lose} = (\cup_i \mathrm{Attr}_i)$; $\mathrm{Win} = \mathrm{Lose}^c$

Finite turn-based safety games



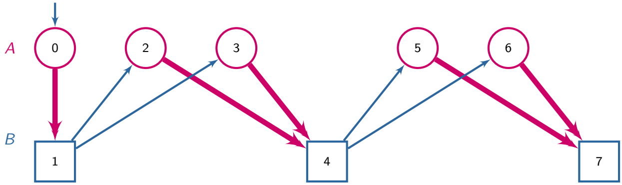
- Strategy of A: $\sigma : V_A \to V_B$
- G_σ: the arena with only moves of σ
- Winning strategy for A: Bad is not reachable in G_σ

Computation of winning and losing states

- Attractors of bad states:
  - $\mathrm{Attr}_0 = \mathrm{Bad}$
  - $\mathrm{Attr}_{i+1} = \mathrm{Attr}_i \cup \{v \in V_B | \exists v' \in \mathrm{Attr}_i, v \to v'\}$
    $\cup \{v \in V_A | \forall v' \in V_B \text{s.t.} v \to v', v' \in \mathrm{Attr}_i\}$
- $\mathrm{Lose} = (\cup_i \mathrm{Attr}_i)$; $\mathrm{Win} = \mathrm{Lose}^c$

$\Rightarrow$ A has a winning strategy iff 0 $\in$ Win

Finite turn-based safety games



- Strategy of A: $\sigma : V_A \to V_B$
- G_σ: the arena with only moves of σ
- Winning strategy for A: Bad is not reachable in G_σ

Computation of winning and losing states

- Attractors of bad states:
  - $\mathrm{Attr}_0 = \mathrm{Bad}$
  - $\mathrm{Attr}_{i+1} = \mathrm{Attr}_i \cup \{v \in V_B \mid \exists v' \in \mathrm{Attr}_i, v \to v'\}$
    $\cup \{v \in V_A \mid \forall v' \in V_B \text{s.t.}\, v \to v', v' \in \mathrm{Attr}_i\}$
- $\mathrm{Lose} = (\cup_i \mathrm{Attr}_i)$; $\mathrm{Win} = \mathrm{Lose}^c$

$\Rightarrow$ A has a winning strategy iff 0 $\in \mathrm{Win}$

Size of strategies

☹ A winning strategy is a huge table ☹

Size of strategies

☹ A winning strategy is a huge table ☹

Observation: some lines are useless to win

- moves from Bad states
- non reachable states in G_σ
- potentially others (*e.g.* states whose all successors are winning)

Size of strategies

☹ A winning strategy is a huge table ☹

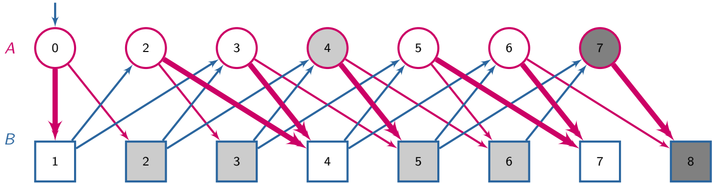
Observation: some lines are useless to win

- moves from Bad states
- non reachable states in G_σ
- potentially others (*e.g.* states whose all successors are winning)

New notion of strategy:

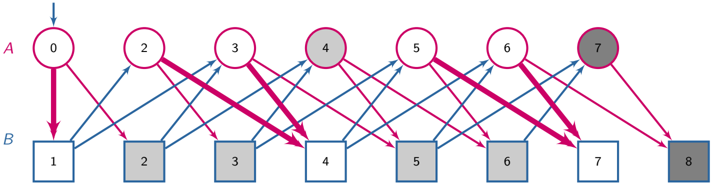
- Strategy of A: $\sigma : V_A \to V_B \cup \{\star\}$ ($\star$ means "don't care")
- G_σ: the arena with only moves of σ (or all the moves if $\sigma(v) = \star$)
- Winning strategy for A: Bad is not reachable in G_σ
- Size of σ: $|\sigma^{-1}(V_B)|$

Back to the running example



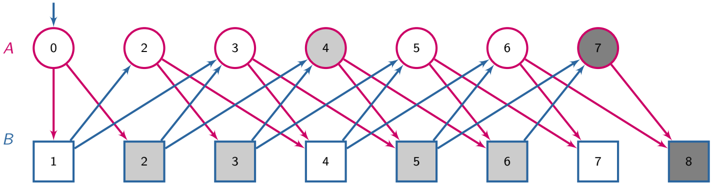
- $|V_A| = 7$
- There is a winning strategy of size 5

Back to the running example



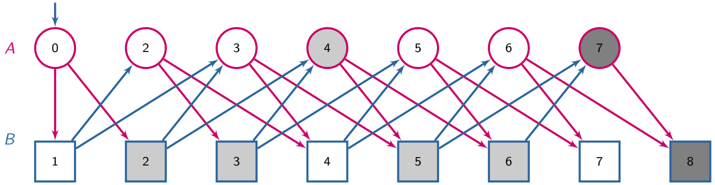
- $|V_A| = 7$
- There is a winning strategy of size 5

Back to the running example



- $|V_A| = 7$
- There is a winning strategy of size 5
- Is there a winning strategy of size 4?

Back to the running example

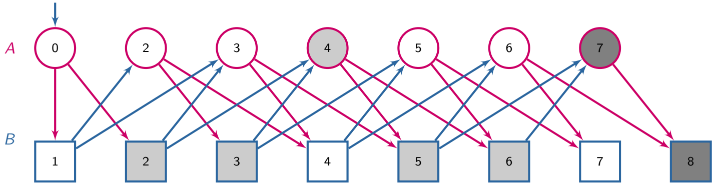


- $|V_A| = 7$
- There is a winning strategy of size 5
- Is there a winning strategy of size 4?

Theorem

Deciding whether there exists a winning strategy for A of size k is an NP-Complete problem.

Back to the running example



- $|V_A| = 7$
- There is a winning strategy of size 5
- Is there a winning strategy of size 4?

Theorem

Deciding whether there exists a winning strategy for A of size k is an NP-Complete problem.

⇒ A heuristic for structured games

Structured games

simulation relation

For all $v_1 \trianglerighteq v_2$, either $v_1 \in \mathrm{Bad}$, or:

- $v_2 \in \mathrm{Bad} \Rightarrow v_1 \in \mathrm{Bad}$
- $v_2 \rightarrow v_2' \Rightarrow v_1 \rightarrow v_1'$ with $v_1' \trianglerighteq v_2'$

Structured games

simulation relation

For all $v_1 \trianglerighteq v_2$, either $v_1 \in \mathrm{Bad}$, or:

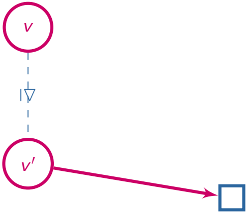
- $v_2 \in \mathrm{Bad} \Rightarrow v_1 \in \mathrm{Bad}$
- $v_2 \to v_2' \Rightarrow v_1 \to v_1'$ with $v_1' \trianglerighteq v_2'$

Structured games

simulation relation

For all $v_1 \trianglerighteq v_2$, either $v_1 \in \mathrm{Bad}$, or:

- $v_2 \in \mathrm{Bad} \Rightarrow v_1 \in \mathrm{Bad}$
- $v_2 \to v_2' \Rightarrow v_1 \to v_1'$ with $v_1' \trianglerighteq v_2'$

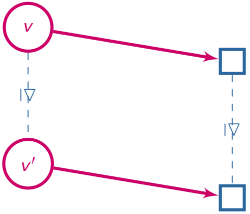


Structured games

simulation relation

For all $v_1 \trianglerighteq v_2$, either $v_1 \in \mathrm{Bad}$, or:

- $v_2 \in \mathrm{Bad} \Rightarrow v_1 \in \mathrm{Bad}$
- $v_2 \to v_2' \Rightarrow v_1 \to v_1'$ with $v_1' \trianglerighteq v_2'$

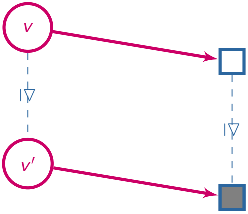


Structured games

simulation relation

For all $v_1 \trianglerighteq v_2$, either $v_1 \in \mathrm{Bad}$, or:

- $v_2 \in \mathrm{Bad} \Rightarrow v_1 \in \mathrm{Bad}$
- $v_2 \to v_2' \Rightarrow v_1 \to v_1'$ with $v_1' \trianglerighteq v_2'$

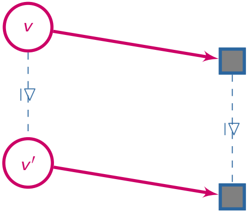


Structured games

simulation relation

For all $v_1 \trianglerighteq v_2$, either $v_1 \in \text{Bad}$, or:

- $v_2 \in \text{Bad} \Rightarrow v_1 \in \text{Bad}$
- $v_2 \to v_2' \Rightarrow v_1 \to v_1'$ with $v_1' \trianglerighteq v_2'$

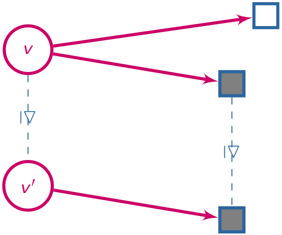


Structured games

simulation relation

For all $v_1 \trianglerighteq v_2$, either $v_1 \in \text{Bad}$, or:

- $v_2 \in \text{Bad} \Rightarrow v_1 \in \text{Bad}$
- $v_2 \to v_2' \Rightarrow v_1 \to v_1'$ with $v_1' \trianglerighteq v_2'$

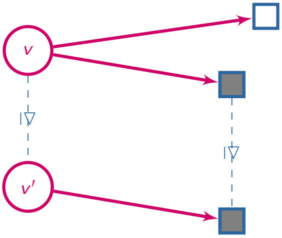


Structured games

A-ready simulation relation

For all $v_1 \trianglerighteq v_2$, either $v_1 \in \mathrm{Bad}$, or:

- $v_2 \in \mathrm{Bad} \;\Rightarrow\; v_1 \in \mathrm{Bad}$
- $v_2 \to v_2' \;\Rightarrow\; v_1 \to v_1'$ with $v_1' \trianglerighteq v_2'$
- $v_1 \to v_1' \;\Rightarrow\; v_2 \to v_2'$ with $v_1' \trianglerighteq v_2'$

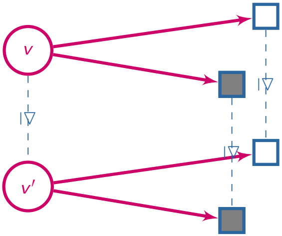


Structured games

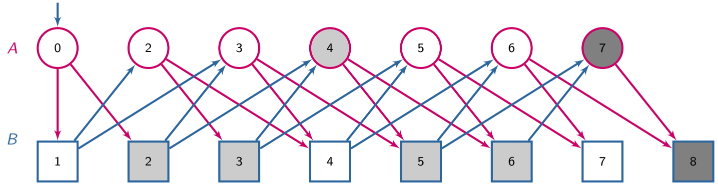
A-ready simulation relation

For all $v_1 \trianglerighteq v_2$, either $v_1 \in \mathrm{Bad}$, or:

- $v_2 \in \mathrm{Bad} \Rightarrow v_1 \in \mathrm{Bad}$
- $v_2 \to v_2' \Rightarrow v_1 \to v_1'$ with $v_1' \trianglerighteq v_2'$
- $v_1 \to v_1' \Rightarrow v_2 \to v_2'$ with $v_1' \trianglerighteq v_2'$



Back to the running example



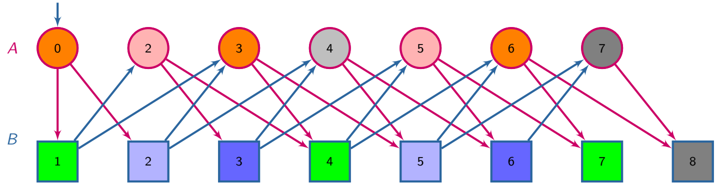
Well-known winning strategy:

always going in a state k of B such that $k \equiv 1 \pmod 3$.

A-ready simulation:

$k \trianglerighteq k'$ & $\boxed{k} \trianglerighteq \boxed{k'}$ if $k \equiv k' \pmod 3$ and $k \geq k'$.

Back to the running example



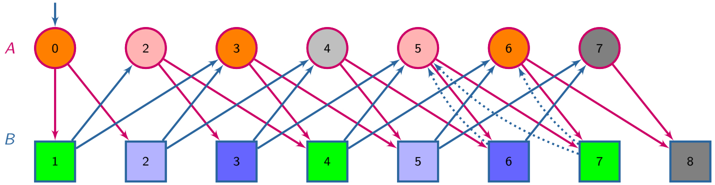
Well-known winning strategy:

always going in a state k of B such that $k \equiv 1 \pmod 3$.

A-ready simulation:

$k \trianglerighteq k'$ & $\boxed{k} \trianglerighteq \boxed{k'}$ if $k \equiv k' \pmod 3$ and $k \geq k'$.

Back to the running example



Well-known winning strategy:

always going in a state k of B such that $k \equiv 1 \pmod 3$.

A-ready simulation:

$k \trianglerighteq k'$ & $\boxed{k} \trianglerighteq \boxed{k'}$ if $k \equiv k' \pmod 3$ and $k \geq k'$.

Succinct strategies

Antichain: S such that $v, v' \in S$ implies that $\neg(v \trianglerighteq v') \wedge \neg(v' \trianglerighteq v)$

Succinct strategies

Antichain: S such that $v, v' \in S$ implies that $\neg(v \trianglerighteq v') \wedge \neg(v' \trianglerighteq v)$

Succinct strategy: $\sigma : V_A \to V_B$ such that $\sigma^{-1}(V_B)$ is an antichain.

Succinct strategies

Antichain: S such that $v, v' \in S$ implies that $\neg(v \trianglerighteq v') \wedge \neg(v' \trianglerighteq v)$

Succinct strategy: $\sigma : V_A \to V_B$ such that $\sigma^{-1}(V_B)$ is an antichain.

Idea: Mimicking the strategy on greater states

New notion of winning strategy

- $G_{\sigma,\trianglerighteq}$: if there are $v' \trianglerighteq v$ where σ is defined, only the moves mimicking such moves
- $\trianglerighteq$-winning strategy: Bad is not reachable in $G_{\sigma,\trianglerighteq}$

Succinct strategies

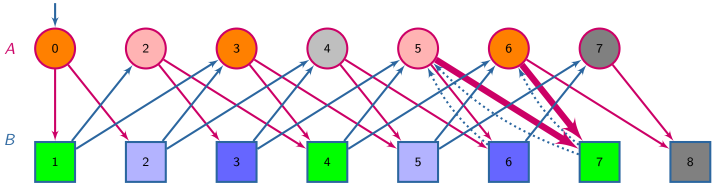
Antichain: S such that $v, v' \in S$ implies that $\neg(v \trianglerighteq v') \wedge \neg(v' \trianglerighteq v)$

Succinct strategy: $\sigma : V_A \to V_B$ such that $\sigma^{-1}(V_B)$ is an antichain.

Idea: Mimicking the strategy on greater states

New notion of winning strategy

- $G_{\sigma,\trianglerighteq}$: if there are $v' \trianglerighteq v$ where σ is defined, only the moves mimicking such moves
- $\trianglerighteq$-winning strategy: Bad is not reachable in $G_{\sigma,\trianglerighteq}$



Succinct strategies

Antichain: S such that $v, v' \in S$ implies that $\neg(v \trianglerighteq v') \wedge \neg(v' \trianglerighteq v)$

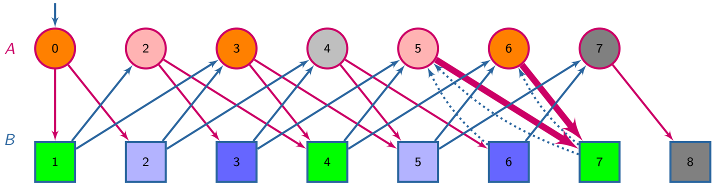
Succinct strategy: $\sigma : V_A \to V_B$ such that $\sigma^{-1}(V_B)$ is an antichain.

Idea: Mimicking the strategy on greater states

New notion of winning strategy

- $G_{\sigma,\trianglerighteq}$: if there are $v' \trianglerighteq v$ where σ is defined, only the moves mimicking such moves
- $\trianglerighteq$-winning strategy: Bad is not reachable in $G_{\sigma,\trianglerighteq}$

G_σ



Succinct strategies

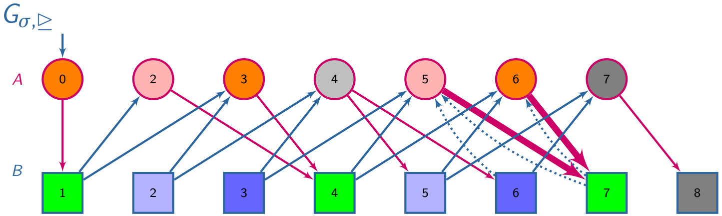
Antichain: S such that $v, v' \in S$ implies that $\neg(v \trianglerighteq v') \wedge \neg(v' \trianglerighteq v)$

Succinct strategy: $\sigma : V_A \to V_B$ such that $\sigma^{-1}(V_B)$ is an antichain.

Idea: Mimicking the strategy on greater states

New notion of winning strategy

- $G_{\sigma,\trianglerighteq}$: if there are $v' \trianglerighteq v$ where σ is defined, only the moves mimicking such moves
- $\trianglerighteq$-winning strategy: Bad is not reachable in $G_{\sigma,\trianglerighteq}$



How to build winning succinct strategy ?

How to build winning succinct strategy ?

Theorem

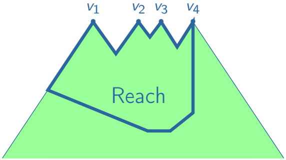
Let σ be a winning strategy and V' be a maximal antichain on $\mathrm{Reach}(G_\sigma) \cap V_A$, then the strategy $\sigma_{|V'}$ is $\trianglerighteq$-winning.

How to build winning succinct strategy ?

Theorem

Let σ be a winning strategy and V' be a maximal antichain on $\mathrm{Reach}(G_\sigma) \cap V_A$, then the strategy $\sigma_{|V'}$ is $\trianglerighteq$-winning.

Intuition:

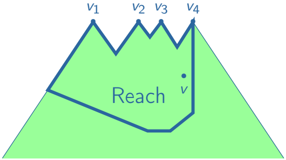


How to build winning succinct strategy ?

Theorem

Let σ be a winning strategy and V' be a maximal antichain on $\mathrm{Reach}(G_\sigma) \cap V_A$, then the strategy $\sigma_{|V'}$ is $\trianglerighteq$-winning.

Intuition:

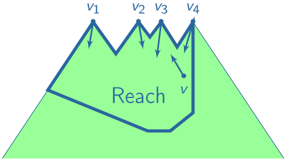


How to build winning succinct strategy ?

Theorem

Let σ be a winning strategy and V' be a maximal antichain on $\text{Reach}(G_\sigma) \cap V_A$, then the strategy $\sigma_{|V'}$ is $\trianglerighteq$-winning.

Intuition:

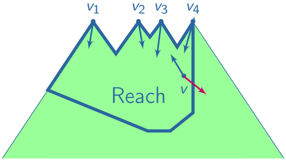


How to build winning succinct strategy ?

Theorem

Let σ be a winning strategy and V' be a maximal antichain on $\mathrm{Reach}(G_\sigma) \cap V_A$, then the strategy $\sigma_{|V'}$ is $\trianglerighteq$-winning.

Intuition:

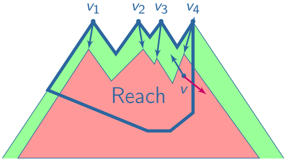


How to build winning succinct strategy ?

Theorem

Let σ be a winning strategy and V' be a maximal antichain on $\mathrm{Reach}(G_\sigma) \cap V_A$, then the strategy $\sigma_{|V'}$ is $\trianglerighteq$-winning.

Intuition:

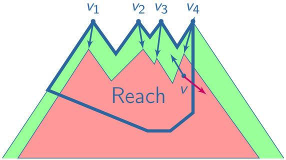


How to build winning succinct strategy ?

Theorem

Let σ be a winning strategy and V' be a maximal antichain on $\text{Reach}(G_\sigma) \cap V_A$, then the strategy $\sigma_{|V'}$ is $\trianglerighteq$-winning.

Intuition:



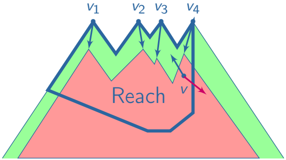
$\Rightarrow$ one can build a $\trianglerighteq$-winning succinct strategy from any winning strategy

How to build winning succinct strategy ?

Theorem

Let σ be a winning strategy and V' be a maximal antichain on $\mathrm{Reach}(G_\sigma) \cap V_A$, then the strategy $\sigma_{|V'}$ is $\trianglerighteq$-winning.

Intuition:



$\Rightarrow$ one can build a $\trianglerighteq$-winning succinct strategy from any winning strategy

v_0	...
v_1	...
v_2	...
v_3	...
...	...
...	...
v_i	...
...	...
v_j	...
...	...
v_{n-1}	...
v_n	...

$\Rightarrow$

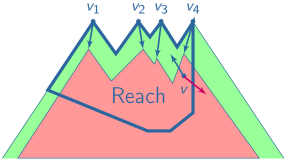
v_0	...
v_i	...
v_j	...
v_{n-1}	...

How to build winning succinct strategy ?

Theorem

Let σ be a winning strategy and V' be a maximal antichain on $\mathrm{Reach}(G_\sigma) \cap V_A$, then the strategy $\sigma_{|V'}$ is $\trianglerighteq$-winning.

Intuition:



$\Rightarrow$ one can build a $\trianglerighteq$-winning succinct strategy from any winning strategy

v_0	...
v_1	...
v_2	...
v_3	...
...	...
...	...
v_i	...
...	...
v_j	...
...	...
v_{n-1}	...
v_n	...

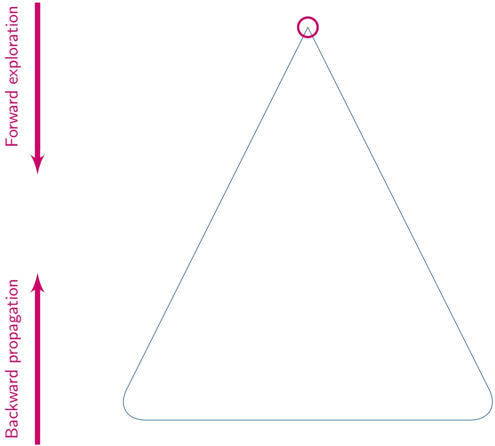
$\Rightarrow$

v_0	...
v_i	...
v_j	...
v_{n-1}	...

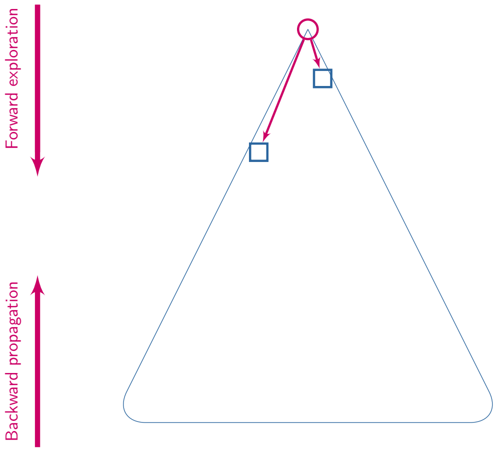
The OTFUR algorithm [CDFLL05] allows to build $\trianglerighteq$-winning succinct strategies.

[CDFLL05] Efficient on-the-fly algorithms for the analysis of timed games.

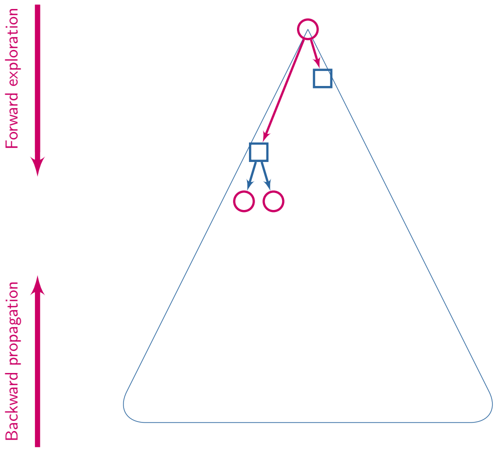
The OTFUR algorithm [CDFLL05]



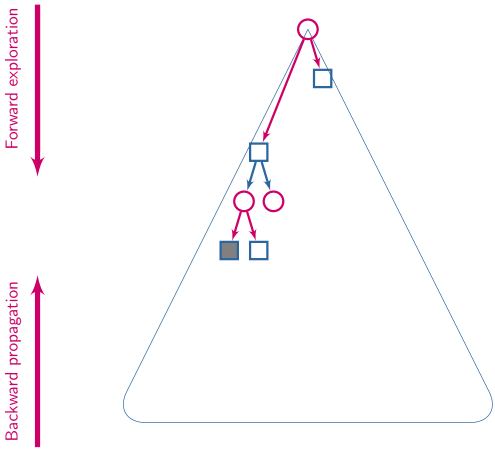
The OTFUR algorithm [CDFLL05]



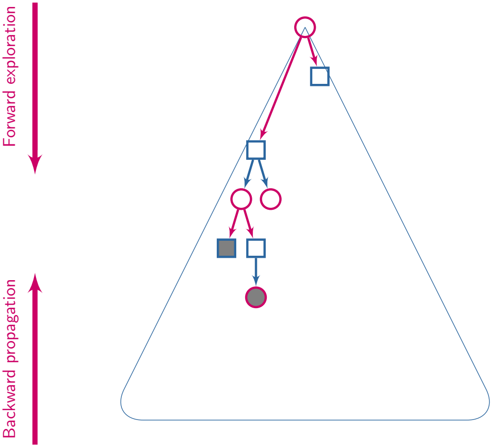
The OTFUR algorithm [CDFLL05]



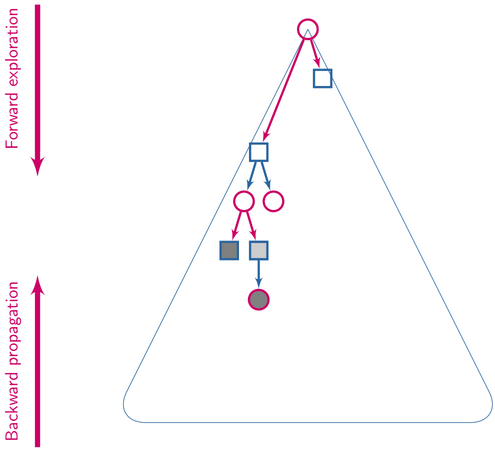
The OTFUR algorithm [CDFLL05]



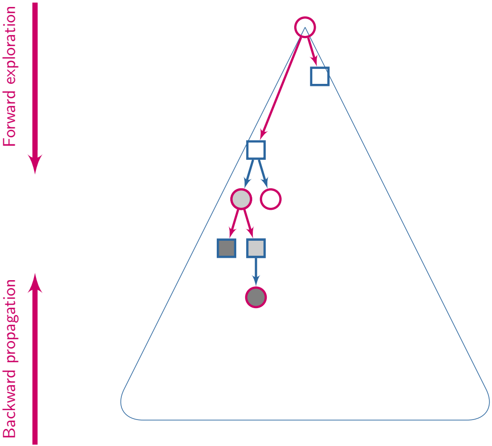
The OTFUR algorithm [CDFLL05]



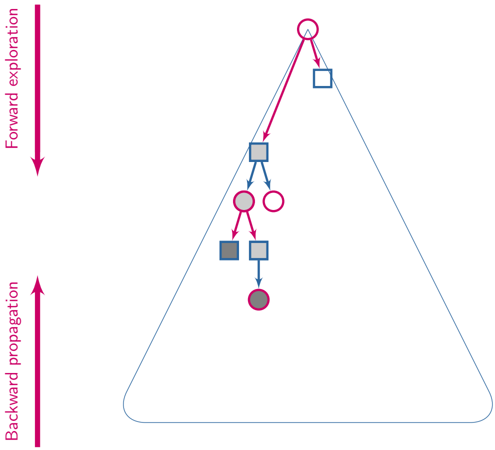
The OTFUR algorithm [CDFLL05]



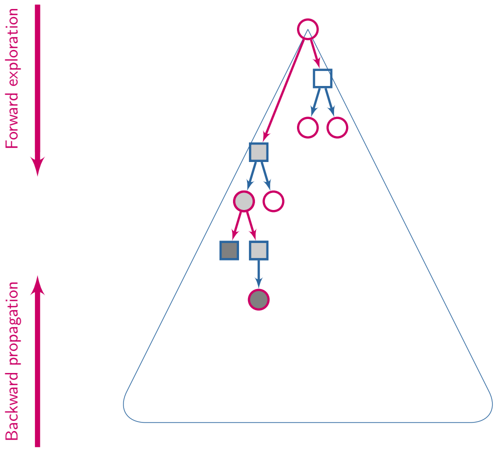
The OTFUR algorithm [CDFLL05]



The OTFUR algorithm [CDFLL05]



The OTFUR algorithm [CDFLL05]



Related work

LTL synthesis [FJR11]: a weaker structure has been identified.

$$\text{simulation relation} + (v \trianglerighteq v' \text{ and } v' \in \mathsf{Lose} \text{ implies } v \in \mathsf{Lose})$$

Related work

LTL synthesis [FJR11]: a weaker structure has been identified.

simulation relation + ($v \trianglerighteq v'$ and $v' \in \mathsf{Lose}$ implies $v \in \mathsf{Lose}$)

- Optimization 1: local optimization

Related work

LTL synthesis [FJR11]: a weaker structure has been identified.

$$\text{simulation relation} + (v \trianglerighteq v' \text{ and } v' \in \text{Lose implies } v \in \text{Lose})$$

- Optimization 1: local optimization

- Optimization 2: speed-up in the detection of losing states

[FJR11] Antichains and compositional algorithms for LTL synthesis.

Related work

LTL synthesis [FJR11]: a weaker structure has been identified.

simulation relation + ($v \trianglerighteq v'$ and $v' \in \text{Lose}$ implies $v \in \text{Lose}$)

- Optimization 1: local optimization

- Optimization 2: speed-up in the detection of losing states

[FJR11] Antichains and compositional algorithms for LTL synthesis.

Related work

LTL synthesis [FJR11]: a weaker structure has been identified.

$$\text{simulation relation} + (v \trianglerighteq v' \text{ and } v' \in \text{Lose implies } v \in \text{Lose})$$

- Optimization 1: local optimization

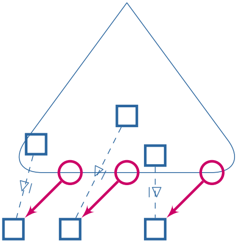
- Optimization 2: speed-up in the detection of losing states

[FJR11] Antichains and compositional algorithms for LTL synthesis.

A new optimization

- Optimization 3: partial exploration of winning states

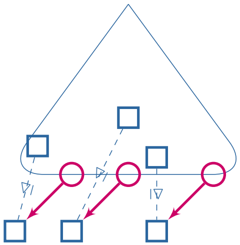
Exploration only
from maximal states



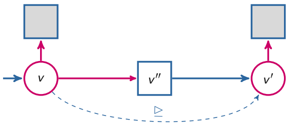
A new optimization

- Optimization 3: partial exploration of winning states

Exploration only
from maximal states



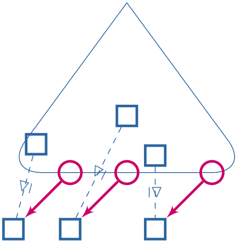
⚠ Not possible with the structure identified in [FJR11]



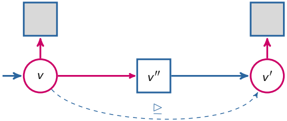
A new optimization

- Optimization 3: partial exploration of winning states

Exploration only
from maximal states

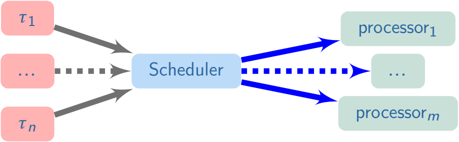


⚠ Not possible with the structure identified in [FJR11]

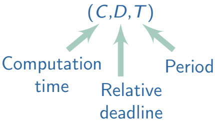


But the relation identified in [FJR11] is an *A*-ready simulation!

Application to real-time scheduling

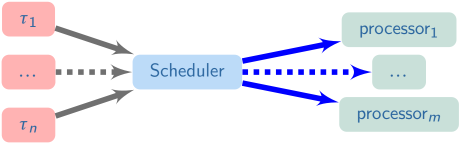


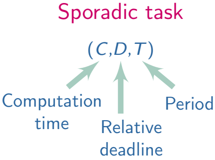
Sporadic task



[BM10] Feasibility Analysis of Sporadic Real-Time Multiprocessor Task Systems.

Application to real-time scheduling

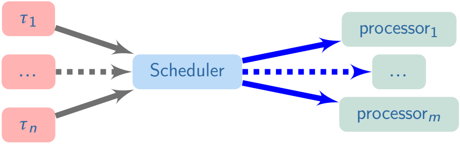


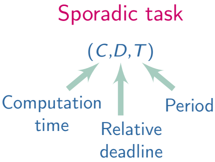


Correctness : no deadline is missed

[BM10] Feasibility Analysis of Sporadic Real-Time Multiprocessor Task Systems.

Application to real-time scheduling





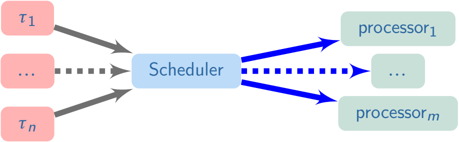
Correctness : no deadline is missed

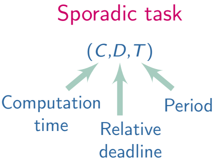
Uniprocessor : optimal algorithms (*e.g.* Earliest Deadline First)

Multiprocessor : no satisfying algorithm in general

[BM10] Feasibility Analysis of Sporadic Real-Time Multiprocessor Task Systems.

Application to real-time scheduling





Correctness : no deadline is missed

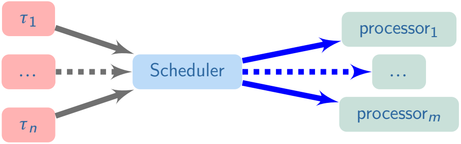
Uniprocessor : optimal algorithms (*e.g.* Earliest Deadline First)

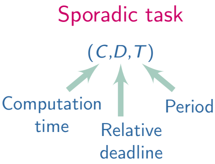
Multiprocessor : no satisfying algorithm in general

- Scheduler synthesis → safety game

[BM10] Feasibility Analysis of Sporadic Real-Time Multiprocessor Task Systems.

Application to real-time scheduling



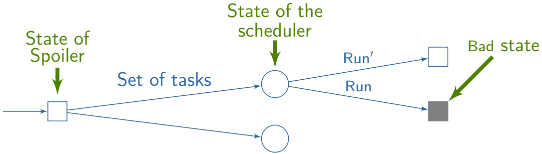


Correctness : no deadline is missed

Uniprocessor : optimal algorithms (*e.g.* Earliest Deadline First)

Multiprocessor : no satisfying algorithm in general

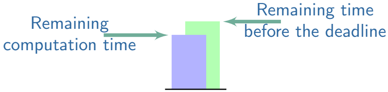
- Scheduler synthesis → safety game



[BM10] Feasibility Analysis of Sporadic Real-Time Multiprocessor Task Systems.

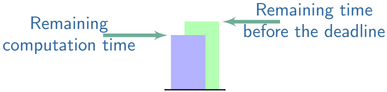
Scheduling game

State of a task:

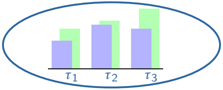


Scheduling game

State of a task:

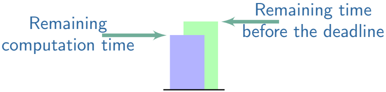


State of the system:

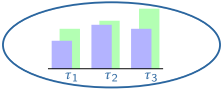


Scheduling game

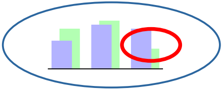
State of a task:



State of the system:

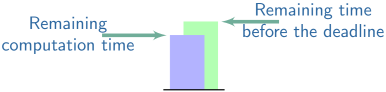


Bad states:

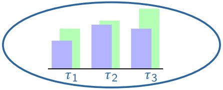


Scheduling game

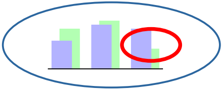
State of a task:



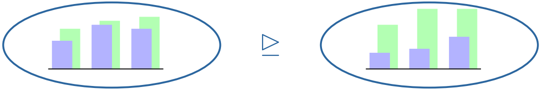
State of the system:



Bad states:



Structure:



How does the approach work?

- Construction of the table

How does the approach work?

- Construction of the table
  - Optimized on-the-fly algorithm to build winning strategies

How does the approach work?

- Construction of the table
  - Optimized on-the-fly algorithm to build winning strategies
  - Extraction of the table of the strategy for maximal states

v_0	...
v_1	...
v_2	...
v_3	...
...	...
...	...
v_i	...
...	...
v_j	...
...	...
v_{n-1}	...
v_n	...

$\Rightarrow$

v_0	...
v_i	...
v_j	...
v_{n-1}	...

How does the approach work?

- Construction of the table
  - Optimized on-the-fly algorithm to build winning strategies
  - Extraction of the table of the strategy for maximal states

v_0	...
v_1	...
v_2	...
v_3	...
...	...
...	...
v_i	...
...	...
v_j	...
...	...
v_{n-1}	...
v_n	...

$\Rightarrow$

v_0	...
v_i	...
v_j	...
v_{n-1}	...

- Functioning of the scheduler

How does the approach work?

- Construction of the table
  - Optimized on-the-fly algorithm to build winning strategies
  - Extraction of the table of the strategy for maximal states

v_0	...
v_1	...
v_2	...
v_3	...
...	...
...	...
v_i	...
...	...
v_j	...
...	...
v_{n-1}	...
v_n	...

$\Rightarrow$

v_0	...
v_i	...
v_j	...
v_{n-1}	...

- Functioning of the scheduler
  - For maximal states: apply the defined strategy

How does the approach work?

- Construction of the table
  - Optimized on-the-fly algorithm to build winning strategies
  - Extraction of the table of the strategy for maximal states

v_0	...
v_1	...
v_2	...
v_3	...
...	...
...	...
v_i	...
...	...
v_j	...
...	...
v_{n-1}	...
v_n	...

$\Rightarrow$

v_0	...
v_i	...
v_j	...
v_{n-1}	...

- Functioning of the scheduler
  - For maximal states: apply the defined strategy
  - For other states: there is at least one greater maximal state

How does the approach work?

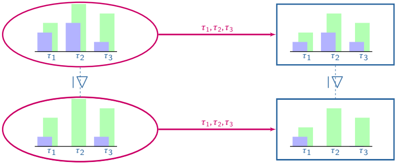
- Construction of the table
  - Optimized on-the-fly algorithm to build winning strategies
  - Extraction of the table of the strategy for maximal states

v_0	...
v_1	...
v_2	...
v_3	...
...	...
...	...
v_i	...
...	...
v_j	...
...	...
v_{n-1}	...
v_n	...

$\Rightarrow$

v_0	...
v_i	...
v_j	...
v_{n-1}	...

- Functioning of the scheduler
  - **For maximal states:** apply the defined strategy
  - **For other states:** there is at least one greater maximal state
    $\Rightarrow$ apply the same priority order



Conclusion

- Contribution
  - Identification of a powerful structure
  - A general approach to define succinct strategies in structured games
  - An optimized on-the-fly algorithm

Conclusion

- Contribution
  - Identification of a powerful structure
  - A general approach to define succinct strategies in structured games
  - An optimized on-the-fly algorithm

- A lot of applications
  - In A-deterministic games, any simulation is an A-ready simulation

Conclusion

- Contribution
  - Identification of a powerful structure
  - A general approach to define succinct strategies in structured games
  - An optimized on-the-fly algorithm

- A lot of applications
  - In A-deterministic games, any simulation is an A-ready simulation
  - Scheduler synthesis

Conclusion

- Contribution
  - Identification of a powerful structure
  - A general approach to define succinct strategies in structured games
  - An optimized on-the-fly algorithm

- A lot of applications
  - In A-deterministic games, any simulation is an A-ready simulation
  - Scheduler synthesis
  - LTL synthesis
    - a new optimization
    - a way to succinctly represent winning strategies

Conclusion

- Contribution
  - Identification of a powerful structure
  - A general approach to define succinct strategies in structured games
  - An optimized on-the-fly algorithm
- A lot of applications
  - In A-deterministic games, any simulation is an A-ready simulation
  - Scheduler synthesis
  - LTL synthesis
    - a new optimization
    - a way to succinctly represent winning strategies
  - Determinization of timed automata

Conclusion

- Contribution
  - Identification of a powerful structure
  - A general approach to define succinct strategies in structured games
  - An optimized on-the-fly algorithm
- A lot of applications
  - In A-deterministic games, any simulation is an A-ready simulation
  - Scheduler synthesis
  - LTL synthesis
    - a new optimization
    - a way to succinctly represent winning strategies
  - Determinization of timed automata
- Current and future work
  - Experimentations using antichains
  - Other applications
  - Weaker relations (more pairs)