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 (~ 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 (~ implementability)
- effective construction

⇒ 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

Strategy of $A$: $\sigma$: $V_A \rightarrow V_B$

Winning strategy for $A$: $\text{Bad}$ is not reachable in $G_{\sigma}$

Computation of winning and losing states:

- Attractors of bad states:
  - $\text{Attr}_0 = \text{Bad}$
  - $\text{Attr}_{i+1} = \text{Attr}_i \cup \{v \in V_B | \exists v' \in \text{Attr}_i, v \rightarrow v'\}$
  - $\text{Attr}_{i+1} = \text{Attr}_i \cup \{v \in V_A | \forall v' \in V_B \text{ s.t. } v \rightarrow v', v' \in \text{Attr}_i\}$

$Lose = \bigcup_i \text{Attr}_i$; $Win = Lose^c$\Rightarrow \quad A$ has a winning strategy iff $0 \in Win$
Finite turn-based safety games

Strategy of $A$: $\sigma : V_A \rightarrow V_B$
Finite turn-based safety games

Strategy of $A$: $\sigma : V_A \rightarrow V_B$

\[ \text{Attr}_0 = \text{Bad} \]
\[ \text{Attr}_{i+1} = \text{Attr}_i \cup \{ v \in V_B \mid \exists v' \in \text{Attr}_i, v \rightarrow v' \} \]
\[ \text{Attr}_{i+1} = \text{Attr}_i \cup \{ v \in V_A \mid \forall v' \in V_B \text{ s.t. } v \rightarrow v', v' \in \text{Attr}_i \} \]

$\mathcal{L} = \bigcup_{i} \text{Attr}_i$; $\mathcal{W} = \mathcal{L}$

$\Rightarrow A$ has a winning strategy iff $0 \in \mathcal{W}$
Finite turn-based safety games

- Strategy of $A$: $\sigma : V_A \rightarrow V_B$
- $G_{\sigma}$: the arena with only moves of $\sigma$
Finite turn-based safety games

Strategy of A: $\sigma : V_A \rightarrow V_B$

$G_\sigma$: the arena with only moves of $\sigma$
Finite turn-based safety games

Strategy of $A$: $\sigma : V_A \rightarrow V_B$

$G_\sigma$: the arena with only moves of $\sigma$

Winning strategy for $A$: Bad is not reachable in $G_\sigma$
Finite turn-based safety games

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

Computation of winning and losing states

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

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

Computation of winning and losing states

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

- **Strategy of A**: $\sigma : V_A \rightarrow V_B$
- $G_\sigma$: the arena with only moves of $\sigma$
- **Winning strategy for A**: Bad is not reachable in $G_\sigma$

**Computation of winning and losing states**

- Attractors of bad states:
  - $Attr_0 = \text{Bad}$
  - $Attr_{i+1} = Attr_i \cup \{v \in V_B | \exists v' \in Attr_i, v \rightarrow v'\}$
    $\cup \{v \in V_A | \forall v' \in V_B \text{s.t. } v \rightarrow v', v' \in Attr_i\}$
- Lose = ($\cup_i Attr_i$); Win = Lose$^c$
Finite turn-based safety games

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

$G_{\sigma}$: the arena with only moves of $\sigma$

Winning strategy for $A$: Bad is not reachable in $G_{\sigma}$

Computation of winning and losing states

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

Strategy of $A$: $\sigma : V_A \rightarrow V_B$

$G_\sigma$: the arena with only moves of $\sigma$

Winning strategy for $A$: Bad is not reachable in $G_\sigma$

Computation of winning and losing states

Attractors of bad states:

- $\text{Attr}_0 = \text{Bad}$
- $\text{Attr}_{i+1} = \text{Attr}_i \cup \{v \in V_B | \exists v' \in \text{Attr}_i, v \rightarrow v'\}
  \cup \{v \in V_A | \forall v' \in V_B \text{s.t.} v \rightarrow v', v' \in \text{Attr}_i\}$

- Lose = $(\cup_i \text{Attr}_i)$; Win = Lose$^c$

$A$ has a winning strategy iff $0 \in \text{Win}$
Finite turn-based safety games

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

Computation of winning and losing states

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

$A$ has a winning strategy iff $0 \in \text{Win}$
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 $G_\sigma$
- 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 $G_\sigma$
- potentially others (e.g. states whose all successors are winning)

New notion of strategy:

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

$\forall |V_A| = 7$

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

\[ |V_A| = 7 \]

\[ \text{There is a winning strategy of size 5} \]

\[ \text{Is there a winning strategy of size 4?} \]

**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 $v_1 \succeq v_2$, either $v_1 \in \text{Bad}$, or:

- $v_2 \in \text{Bad} \implies v_1 \in \text{Bad}$
- $v_2 \rightarrow v'_2 \implies v_1 \rightarrow v'_1$ with $v'_1 \succeq v'_2$
Structured games

Simulation relation

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

- $v_2 \in \text{Bad} \implies v_1 \in \text{Bad}$
- $v_2 \rightarrow v_2' \implies v_1 \rightarrow v_1'$ with $v_1' \succeq v_2'$
Structured games

**simulation relation**

For all $v_1 \supseteq v_2$, 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 $v'_1 \supseteq v'_2$
For all $v_1 \triangleright v_2$, either $v_1 \in \text{Bad}$, or:

- $v_2 \in \text{Bad} \Rightarrow v_1 \in \text{Bad}$
- $v_2 \rightarrow v'_2 \Rightarrow v_1 \rightarrow v'_1$ with $v'_1 \triangleright v'_2$
Structured games

Simulation relation

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

- $v_2 \in \text{Bad} \implies v_1 \in \text{Bad}$
- $v_2 \rightarrow v'_2 \implies v_1 \rightarrow v'_1$ with $v'_1 \triangleright v'_2$
Structured games

Simulation relation

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

- $v_2 \in \text{Bad} \Rightarrow v_1 \in \text{Bad}$
- $v_2 \rightarrow v_2' \Rightarrow v_1 \rightarrow v_1'$ with $v_1' \triangleright= v_2'$
Structured games

Simulation relation

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

- $v_2 \in \text{Bad} \Rightarrow v_1 \in \text{Bad}$
- $v_2 \rightarrow v_2' \Rightarrow v_1 \rightarrow v_1'$ with $v_1' \geq v_2'$
Structured games

A-ready simulation relation

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

- $v_2 \in \text{Bad} \Rightarrow v_1 \in \text{Bad}$
- $v_2 \rightarrow v'_2 \Rightarrow v_1 \rightarrow v'_1$ with $v'_1 \triangleright v'_2$
- $v_1 \rightarrow v'_1 \Rightarrow v_2 \rightarrow v'_2$ with $v'_1 \triangleright v'_2$
Structured games

A-ready simulation relation

For all \( v_1 \triangleright v_2 \), either
\[ v_1 \in \text{Bad}, \]
or:
\[ v_2 \in \text{Bad} \Rightarrow v_1 \in \text{Bad} \]
\[ v_2 \rightarrow v'_2 \Rightarrow v_1 \rightarrow v'_1 \text{ with } v'_1 \triangleright v'_2 \]
\[ v_1 \rightarrow v'_1 \Rightarrow v_2 \rightarrow v'_2 \text{ with } v'_1 \triangleright v'_2 \]
Well-known winning strategy:

always going in a state \( k \) of \( B \) such that \( k \equiv 1 \pmod{3} \).

\( A \)-ready simulation:

\[
\begin{align*}
\text{circle} & \quad \geq \quad \text{circle} & \\
\text{square} & \quad \geq \quad \text{square} & \quad \text{if } k \equiv k' \pmod{3} \text{ and } k \geq k'.
\end{align*}
\]
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:

$k \triangleright k'$ & $k \triangleright k'$ 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:

$k \blacklozenge k' \quad \& \quad k \blacktriangleleft k'$ if $k \equiv k' \pmod{3}$ and $k \geq k'$.
Succinct strategies

Antichain: $S$ such that $v, v' \in S$ implies that $\neg (v \triangleright v') \land \neg (v' \triangleright v)$
Succinct strategies

Antichain: $S$ such that $v, v' \in S$ implies that $\neg(v \triangleright v') \land \neg(v' \triangleright v)$

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

Antichain: $S$ such that $v, v' \in S$ implies that $\neg(v \triangleright v') \land \neg(v' \triangleright v)$

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

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

**Antichain:** $S$ such that $v, v' \in S$ implies that $\neg(v \triangleright v') \land \neg(v' \triangleright v)$

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

**Idea:** Mimicking the strategy on greater states

**New notion of winning strategy**
- $G_{\sigma,\triangleright}$: if there are $v' \triangleright v$ where $\sigma$ is defined, only the moves mimicking such moves
- $\triangleright$-winning strategy: Bad is not reachable in $G_{\sigma,\triangleright}$
Succinct strategies

Antichain: $S$ such that $v, v' \in S$ implies that $\neg(v \triangleright v') \land \neg(v' \triangleright v)$

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

Idea: Mimicking the strategy on greater states

New notion of winning strategy
- $G_{\sigma, \triangleright}$: if there are $v' \triangleright v$ where $\sigma$ is defined, only the moves mimicking such moves
- $\triangleright$-winning strategy: Bad is not reachable in $G_{\sigma, \triangleright}$
Succinct strategies

Antichain: $S$ such that $v, v' \in S$ implies that $\neg (v \triangleright v') \land \neg (v' \triangleright v)$

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

- $G_{\sigma, \triangleright}$: if there are $v' \triangleright v$ where $\sigma$ is defined, only the moves mimicking such moves
- $\triangleright$-winning strategy: Bad is not reachable in $G_{\sigma, \triangleright}$
How to build winning succinct strategy?

Theorem

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

Intuition:

Reach $v_1 \rightarrow v_2 \rightarrow v_3 \rightarrow v_4 \Rightarrow$ one can build a $\tau$-winning succinct strategy from any winning strategy $v_0 \ldots v_1 \ldots v_2 \ldots v_3 \ldots \ldots \ldots v_i \ldots v_j \ldots \ldots v_n - 1 \ldots v_n \Rightarrow$

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

[CDFLL05] Efficient on-the-fly algorithms for the analysis of timed games.
How to build winning succinct strategy?

**Theorem**

Let $\sigma$ be a winning strategy and $V'$ be a maximal antichain on $\text{Reach}(G_\sigma) \cap V_A$, then the strategy $\sigma|_{V'}$ is $\succeq$-winning.
How to build winning succinct strategy?

**Theorem**

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

**Intuition:**

One can build a $\lhd$-winning succinct strategy from any winning strategy $v_0 \ldots v_i \ldots v_j \ldots v_n - 1 \ldots v_n$.
How to build winning succinct strategy?

Theorem

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

Intuition:
How to build winning succinct strategy?

**Theorem**

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

**Intuition:**

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

[Amélie Stainer](#)  
Grenoble – April 2014 10/17
How to build winning succinct strategy?

**Theorem**

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

**Intuition:**

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

[28x254]How to build winning succinct strategy ?

Theorem

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

Intuition:

The OTFUR algorithm [CDFLL05] allows to build $\triangleright$-winning succinct strategies.
How to build winning succinct strategy?

Theorem

Let $\sigma$ 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:

Reach

$v_1$ $v_2$ $v_3$ $v_4$
Theorem

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

Intuition:

⇒ one can build a $\supseteq$-winning succinct strategy from any winning strategy
How to build winning succinct strategy?

**Theorem**

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

**Intuition:**

$\Rightarrow$ one can build a $\triangleright$-winning succinct strategy from any winning strategy.
How to build winning succinct strategy?

Theorem

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

Intuition:

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

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

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

Forward exploration

Backward propagation
The OTFUR algorithm [CDFLL05]
The OTFUR algorithm [CDFLL05]
The OTFUR algorithm [CDFLL05]

Forward exploration

Backward propagation
The OTFUR algorithm [CDFLL05]
The OTFUR algorithm [CDFLL05]
The OTFUR algorithm [CDFLL05]

Forward exploration

Backward propagation
The OTFUR algorithm [CDFLL05]
The OTFUR algorithm [CDFLL05]
Related work

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

\[ v \trianglerighteq v' \text{ and } v' \in \text{Lose implies } v \in \text{Lose} \]
Related work

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

\[ \text{simulation relation } + (v \triangleright v' \text{ and } v' \in \text{Lose implies } v \in \text{Lose}) \]

- **Optimization 1:** local optimization

![Diagram showing local optimization]
Related work

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

\[
\text{simulation relation } + (v \geq v' \text{ and } v' \in \text{Lose implies } v \in \text{Lose})
\]

- **Optimization 1**: local optimization

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

Related work

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

\[ \text{simulation relation} + (v \geq v' \text{ and } v' \in \text{Lose implies } v \in \text{Lose}) \]

- **Optimization 1:** local optimization

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

Related work

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

\[
\text{simulation relation } + (v \succeq v' \text{ and } v' \in \text{Lose implies } v \in \text{Lose})
\]

- **Optimization 1:** local optimization

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

A new optimization

- **Optimization 3**: partial exploration of winning states

Exploration only from maximal states
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] $(C,D,T)$

Computation time  Relative deadline  Period

Scheduler synthesis $\rightarrow$ safety game

State of Spoiler scheduler  State of the Bad state

Set of tasks  Run $\rightarrow$ Run

Feasibility Analysis of Sporadic Real-Time Multiprocessor Task Systems.

Application to real-time scheduling

Correctness: no deadline is missed

Application to real-time scheduling

Correctness: no deadline is missed

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

Multiprocessor: no satisfying algorithm in general

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 $\rightarrow$ safety game

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

Scheduling game

State of a task:

Remaining computation time

Remaining time before the deadline
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:

Remaining computation time

Remaining time before the deadline

State of the system:

$\tau_1$ $\tau_2$ $\tau_3$

Bad states:

Structure:
How does the approach work?

- Construction of the table

Functioning of the scheduler

- For maximal states: apply the defined strategy
- For other states: there is at least one greater maximal state ⇒ apply the same priority order

$v_0 \ldots v_i \ldots v_j \ldots v_{n-1} \ldots v_n$
How does the approach work?

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

- Functioning of the scheduler
  - For maximal states: apply the defined strategy
  - For other states: there is at least one greater maximal state
    \[ \tau_1 \tau_2 \tau_3 \Rightarrow \tau_1 \tau_2 \tau_3 \]
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

\[
\begin{array}{cccc}
\vdots \\
v_0 \\
v_1 \\
v_2 \\
v_3 \\
\vdots \\
\vdots \\
v_j \\
\vdots \\
v_{n-1} \\
v_n \\
\end{array}
\Rightarrow
\begin{array}{cccc}
\vdots \\
v_0 \\
v_1 \\
v_2 \\
v_3 \\
\vdots \\
\vdots \\
v_j \\
\vdots \\
v_{n-1} \\
v_n \\
\end{array}
\]
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

- 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

\[
\begin{array}{c}
\nu_0 \ldots \\
\nu_1 \ldots \\
\nu_2 \ldots \\
\vdots \ldots \\
\nu_i \ldots \\
\vdots \ldots \\
\nu_{n-1} \ldots \\
\nu_n \ldots \\
\end{array}
\Rightarrow
\begin{array}{c}
\nu_0 \ldots \\
\nu_i \ldots \\
\nu_j \ldots \\
\nu_{n-1} \ldots \\
\end{array}
\]

- **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 ...
```

⇒

```
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 ...
```

⇒

```

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

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

⇒ 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)