Sky Is Not the Limit:

Tighter Rank Bounds for Elevator Automata in Büchi Automata Complementation

Vojtěch Havlena **Ondřej Lengál** Barbora Šmahlíková

Brno University of Technology, Czech Republic

TACAS'22

Büchi Automata

Büchi automata (BAs):

- Automata over infinite words
- $\mathcal{A} = (Q, \delta, I, Acc)$ over Σ
  - Q finite set of states
  - δ transition relation; $\delta \subseteq Q \times \Sigma \times Q$
  - $I \subseteq Q$ initial states
  - $Acc \subseteq Q$ accepting states

Büchi Automata

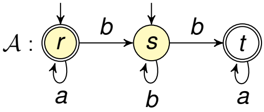
Büchi automata (BAs):

- Automata over infinite words
- $\mathcal{A} = (Q, \delta, I, Acc)$ over Σ
  - Q finite set of states
  - δ transition relation; $\delta \subseteq Q \times \Sigma \times Q$
  - $I \subseteq Q$ initial states
  - $Acc \subseteq Q$ accepting states
- accept by looping over accepting states

Büchi Automata

Büchi automata (BAs):

- Automata over infinite words
- $\mathcal{A} = (Q, \delta, I, Acc)$ over Σ
  - Q finite set of states
  - δ transition relation; $\delta \subseteq Q \times \Sigma \times Q$
  - $I \subseteq Q$ initial states
  - $Acc \subseteq Q$ accepting states
- accept by looping over accepting states

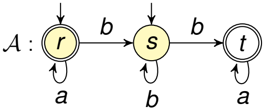


Büchi Automata

Büchi automata (BAs):

- Automata over infinite words
- $\mathcal{A} = (Q, \delta, I, Acc)$ over Σ
  - Q finite set of states
  - δ transition relation; $\delta \subseteq Q \times \Sigma \times Q$
  - $I \subseteq Q$ initial states
  - $Acc \subseteq Q$ accepting states

- accept by looping over accepting states



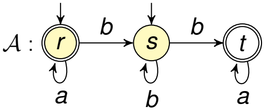
- $r \xrightarrow{a} r \xrightarrow{b} s \xrightarrow{b} t \xrightarrow{a} t \xrightarrow{a} \cdots \quad abba^\omega \in \mathcal{L}(\mathcal{A})$

Büchi Automata

Büchi automata (BAs):

- Automata over infinite words
- $\mathcal{A} = (Q, \delta, I, Acc)$ over Σ
  - Q finite set of states
  - δ transition relation; $\delta \subseteq Q \times \Sigma \times Q$
  - $I \subseteq Q$ initial states
  - $Acc \subseteq Q$ accepting states

- accept by looping over accepting states



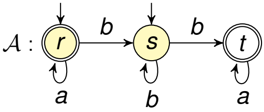
- $r \xrightarrow{a} r \xrightarrow{b} s \xrightarrow{b} t \xrightarrow{a} t \xrightarrow{a} \cdots \quad abba^\omega \in \mathcal{L}(\mathcal{A})$
- $L(\mathcal{A}) = (\epsilon + a^* bb^+ + b^+)a^\omega$

Büchi Automata

Büchi automata (BAs):

- Automata over infinite words
- $\mathcal{A} = (Q, \delta, I, Acc)$ over Σ
  - Q finite set of states
  - δ transition relation; $\delta \subseteq Q \times \Sigma \times Q$
  - $I \subseteq Q$ initial states
  - $Acc \subseteq Q$ accepting states

- accept by looping over accepting states



- $r \xrightarrow{a} r \xrightarrow{b} s \xrightarrow{b} t \xrightarrow{a} t \xrightarrow{a} \cdots \qquad abba^\omega \in \mathcal{L}(\mathcal{A})$
- $L(\mathcal{A}) = (\epsilon + a^* bb^+ + b^+)a^\omega$

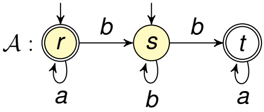
- define the class of ω-regular languages

Büchi Automata

Büchi automata (BAs):

- Automata over infinite words
- $\mathcal{A} = (Q, \delta, I, Acc)$ over Σ
  - Q finite set of states
  - δ transition relation; $\delta \subseteq Q \times \Sigma \times Q$
  - $I \subseteq Q$ initial states
  - $Acc \subseteq Q$ accepting states

- accept by looping over accepting states



- $r \xrightarrow{a} r \xrightarrow{b} s \xrightarrow{b} t \xrightarrow{a} t \xrightarrow{a} \cdots \quad abba^\omega \in \mathcal{L}(\mathcal{A})$
- $L(\mathcal{A}) = (\epsilon + a^* bb^+ + b^+)a^\omega$

- define the class of ω-regular languages
- used in program verification (Ultimate Automizer), linear time MC, probabilistic MC, decision procedures, . . .

BA Complementation

Complementation:

- Given $\mathcal{A}$, get a BA $\mathcal{A}^{\complement}$ such that $\mathcal{L}(\mathcal{A}^{\complement}) = \overline{\mathcal{L}(\mathcal{A})}$.

BA Complementation

Complementation:

- Given $\mathcal{A}$, get a BA $\mathcal{A}^{\complement}$ such that $\mathcal{L}(\mathcal{A}^{\complement}) = \overline{\mathcal{L}(\mathcal{A})}$.

Motivation:

- Model checking of linear-time properties

$$\underbrace{S}_{\text{system}} \models \underbrace{\varphi}_{\text{property}} \rightsquigarrow \quad \mathcal{L}(\mathcal{A}_S) \subseteq \mathcal{L}(\mathcal{A}_\varphi) \quad \rightsquigarrow \quad \mathcal{L}(\mathcal{A}_S) \cap \mathcal{L}(\mathcal{A}_\varphi^{\complement}) = \emptyset$$

BA Complementation

Complementation:

- Given $\mathcal{A}$, get a BA $\mathcal{A}^{\complement}$ such that $\mathcal{L}(\mathcal{A}^{\complement}) = \overline{\mathcal{L}(\mathcal{A})}$.

Motivation:

- Model checking of linear-time properties

$$\underbrace{S}_{\text{system}} \models \underbrace{\varphi}_{\text{property}} \rightsquigarrow \mathcal{L}(\mathcal{A}_S) \subseteq \mathcal{L}(\mathcal{A}_\varphi) \rightsquigarrow \mathcal{L}(\mathcal{A}_S) \cap \mathcal{L}(\mathcal{A}_\varphi^{\complement}) = \emptyset$$

- Termination analysis of programs: Ultimate Automizer
  - removing traces with proved termination
  - difference automaton

BA Complementation

Complementation:

- Given $\mathcal{A}$, get a BA $\mathcal{A}^{\complement}$ such that $\mathcal{L}(\mathcal{A}^{\complement}) = \overline{\mathcal{L}(\mathcal{A})}$.

Motivation:

- Model checking of linear-time properties

$$\underbrace{S}_{\text{system}} \models \underbrace{\varphi}_{\text{property}} \rightsquigarrow \quad \mathcal{L}(\mathcal{A}_S) \subseteq \mathcal{L}(\mathcal{A}_\varphi) \quad \rightsquigarrow \quad \mathcal{L}(\mathcal{A}_S) \cap \mathcal{L}(\mathcal{A}_\varphi^{\complement}) = \emptyset$$

- Termination analysis of programs: Ultimate Automizer
  - removing traces with proved termination
  - difference automaton
- Decision procedures: implements negation
  - S1S: MSO over $(\omega, 0, +1)$
  - QPTL: quantified propositional temporal logic
  - FO over Sturmian words

BA Complementation

Complementation:

- Given $\mathcal{A}$, get a BA $\mathcal{A}^{\complement}$ such that $\mathcal{L}(\mathcal{A}^{\complement}) = \overline{\mathcal{L}(\mathcal{A})}$.

Motivation:

- Model checking of linear-time properties

$$\underbrace{S}_{\text{system}} \models \underbrace{\varphi}_{\text{property}} \rightsquigarrow \quad \mathcal{L}(\mathcal{A}_S) \subseteq \mathcal{L}(\mathcal{A}_\varphi) \quad \rightsquigarrow \quad \mathcal{L}(\mathcal{A}_S) \cap \mathcal{L}(\mathcal{A}_\varphi^{\complement}) = \emptyset$$

- Termination analysis of programs: Ultimate Automizer
  - removing traces with proved termination
  - difference automaton
- Decision procedures: implements negation
  - S1S: MSO over $(\omega, 0, +1)$
  - QPTL: quantified propositional temporal logic
  - FO over Sturmian words
- Basic operation for inclusion/equivalence checking

BA Complementation

Complementation:

- Given $\mathcal{A}$, get a BA $\mathcal{A}^{\complement}$ such that $\mathcal{L}(\mathcal{A}^{\complement}) = \overline{\mathcal{L}(\mathcal{A})}$.

Motivation:

- Model checking of linear-time properties

$$\underbrace{S}_{\text{system}} \models \underbrace{\varphi}_{\text{property}} \rightsquigarrow \quad \mathcal{L}(\mathcal{A}_S) \subseteq \mathcal{L}(\mathcal{A}_\varphi) \quad \rightsquigarrow \quad \mathcal{L}(\mathcal{A}_S) \cap \mathcal{L}(\mathcal{A}_\varphi^{\complement}) = \emptyset$$

- Termination analysis of programs: Ultimate Automizer
  - removing traces with proved termination
  - difference automaton
- Decision procedures: implements negation
  - S1S: MSO over $(\omega, 0, +1)$
  - QPTL: quantified propositional temporal logic
  - FO over Sturmian words
- Basic operation for inclusion/equivalence checking
- Beautiful and ☺fun☺!

BA Complementation

- Notoriously difficult...
  - exponential worst-case lower bound $(0.76n)^n$ [Yan'06]

BA Complementation

- Notoriously difficult…
  - exponential worst-case lower bound $(0.76n)^n$ [Yan'06]

Approaches:

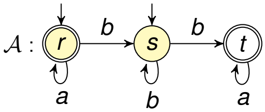
- Ramsey-based [Sistla,Vardi,Volper'87][BreuersLO'12]
- Determinization-based (SPOT, LTL2DSTAR) [Safra'88][Piterman'06][Redziejowski12]
- Slice-based [Vardi,Wilke'08][Kähler,Wilke'08]
- Learning-based [Li,Turrini,Zhang,Schewe'18]
- Subset-tuple construction [Allred,Utes-Nitche'18]
- Semideterminization-based (SEMINATOR 2) [BlahoudekDS'20]
- **Rank-based** [KupfermanV'01][FriedgutKV'06][Schewe'09]

Rank-based Complementation of Büchi Automata

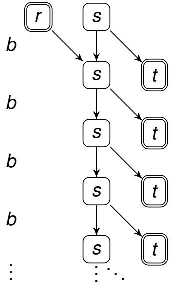
- [Kupferman & Vardi 2001]
- [Friedgut, Kupferman & Vardi 2006]
- [Schewe 2009]
- [Chen, Havlena & L. 2019]
- [Havlena & L. 2021]
- this talk

Rank-based Complementation

- Run DAG $\mathcal{G}_w$ of $\mathcal{A}$ on the word w
  - represents all runs of $\mathcal{A}$ on w
  - $w \notin \mathcal{L}(\mathcal{A})$ iff no ∞ accepting path

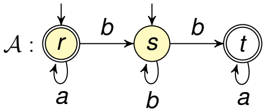


run DAG for $b^\omega \notin \mathcal{L}(\mathcal{A})$



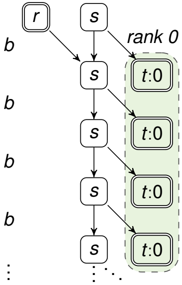
Rank-based Complementation

- Run DAG $\mathcal{G}_w$ of $\mathcal{A}$ on the word w
  - represents all runs of $\mathcal{A}$ on w
  - $w \notin \mathcal{L}(\mathcal{A})$ iff no ∞ accepting path

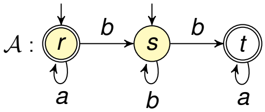


- Ranking procedure (start with $i = 0$)
  1. assign rank i to vertices with finitely many successors and remove them from $\mathcal{G}_w$

run DAG for $b^\omega \notin \mathcal{L}(\mathcal{A})$

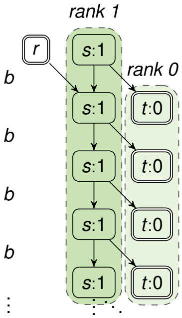


- Run DAG $\mathcal{G}_w$ of $\mathcal{A}$ on the word w
  - represents all runs of $\mathcal{A}$ on w
  - $w \notin \mathcal{L}(\mathcal{A})$ iff no ∞ accepting path

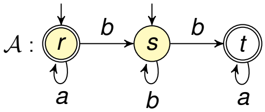


- Ranking procedure (start with $i = 0$)
  1. assign rank i to vertices with finitely many successors and remove them from $\mathcal{G}_w$
  2. assign rank $i + 1$ to vertices that cannot reach *Acc* and remove them from $\mathcal{G}_w$

run DAG for $b^\omega \notin \mathcal{L}(\mathcal{A})$



Rank-based Complementation

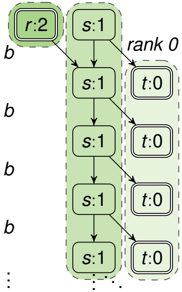


- Run DAG $\mathcal{G}_w$ of $\mathcal{A}$ on the word w
  - represents all runs of $\mathcal{A}$ on w
  - $w \notin \mathcal{L}(\mathcal{A})$ iff no ∞ accepting path

- Ranking procedure (start with $i = 0$)
  1. assign rank i to vertices with finitely many successors and remove them from $\mathcal{G}_w$
  2. assign rank $i + 1$ to vertices that cannot reach *Acc* and remove them from $\mathcal{G}_w$
  3. $i := i + 2$;

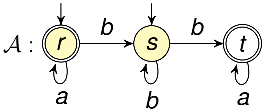
     repeat until $\mathcal{G}_w = \emptyset$

run DAG for $b^\omega \notin \mathcal{L}(\mathcal{A})$



Rank-based Complementation

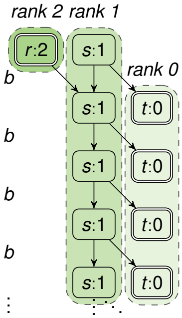
- Run DAG $\mathcal{G}_w$ of $\mathcal{A}$ on the word w
  - represents all runs of $\mathcal{A}$ on w
  - $w \notin \mathcal{L}(\mathcal{A})$ iff no ∞ accepting path



- Ranking procedure (start with $i = 0$)
  1. assign rank i to vertices with finitely many successors and remove them from $\mathcal{G}_w$
  2. assign rank $i + 1$ to vertices that cannot reach *Acc* and remove them from $\mathcal{G}_w$
  3. $i := i + 2$;

     repeat until $\mathcal{G}_w = \emptyset$

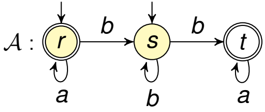
run DAG for $b^\omega \notin \mathcal{L}(\mathcal{A})$

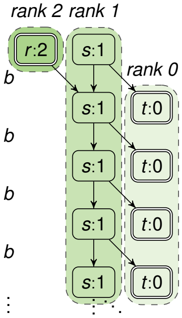


Lemma [Kupferman, Vardi'01]

$w \notin \mathcal{L}(\mathcal{A}) \quad \Leftrightarrow \quad \forall v: \mathit{rank}(v) \leq 2|Q|$

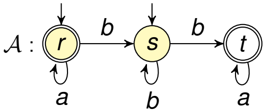
Rank-based Complementation

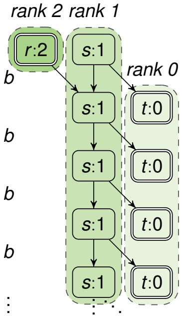




run DAG of $\mathcal{A}$ over b^ω

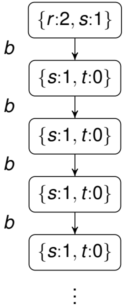
Rank-based Complementation





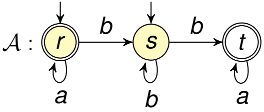
run DAG of $\mathcal{A}$ over b^ω

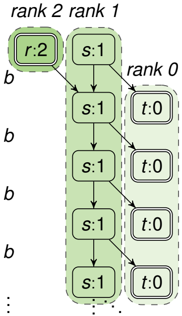
$\rightsquigarrow$



run of $\mathcal{A}^\complement$ over b^ω

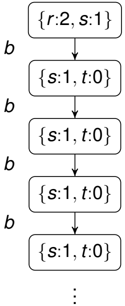
Rank-based Complementation





run DAG of $\mathcal{A}$ over b^ω

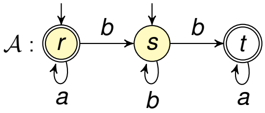
$\rightsquigarrow$

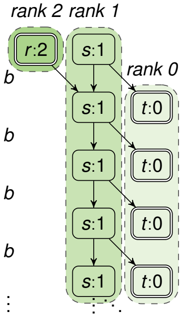


run of $\mathcal{A}^\complement$ over b^ω

- track all runs

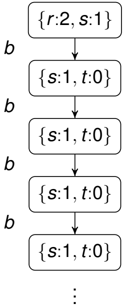
Rank-based Complementation





run DAG of $\mathcal{A}$ over b^ω

$\rightsquigarrow$

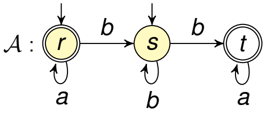


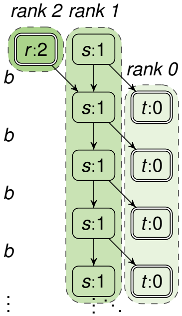
run of $\mathcal{A}^\complement$ over b^ω

- track all runs

Guess & Check:

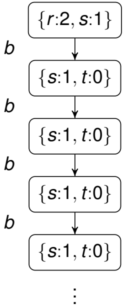
Rank-based Complementation





run DAG of $\mathcal{A}$ over b^ω

$\rightsquigarrow$



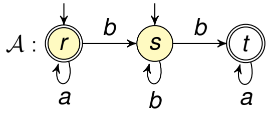
run of $\mathcal{A}^\complement$ over b^ω

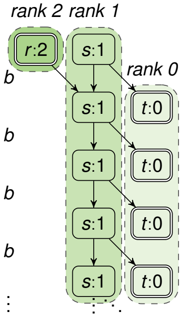
- track all runs

Guess & Check:

- nondet. choice of ranking: $\{r{:}0, s{:}1\}, \{r{:}2, s{:}0\}, \ldots$

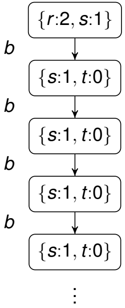
Rank-based Complementation





run DAG of $\mathcal{A}$ over b^ω

$\rightsquigarrow$



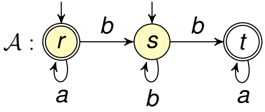
run of $\mathcal{A}^\complement$ over b^ω

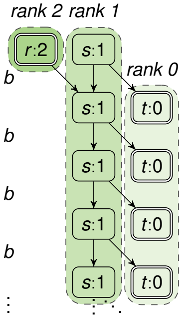
- track all runs

Guess & Check:

- nondet. choice of ranking: $\{r{:}0, s{:}1\}, \{r{:}2, s{:}0\}, \ldots$
- check no run ∞ *Acc*
  - breakpoint

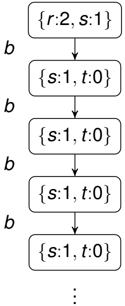
Rank-based Complementation





run DAG of $\mathcal{A}$ over b^ω

$\rightsquigarrow$



run of $\mathcal{A}^\complement$ over b^ω

- track all runs

Guess & Check:

- nondet. choice of ranking: $\{r{:}0, s{:}1\}, \{r{:}2, s{:}0\}, \ldots$
- check no run ∞ *Acc*
  - breakpoint
- ranks on runs can never increase

Rank-based Complementation *Problems*

Source of state explosion:

- size of $\mathcal{A}^{\complement}$ depends on the factorial of the rank bound
  - the maximum finite rank of $\mathcal{G}_w$ for $w \notin \mathcal{L}(\mathcal{A})$
  - e.g., $\{q, r, s, t\}$, $bound = 5$, $\rightsquigarrow$
    - $\{q{:}5, r{:}5, s{:}3, t{:}1\}, \{q{:}3, r{:}3, s{:}1, t{:}5\}, \{q{:}1, r{:}3, s{:}1, t{:}5\}, \ldots$

Rank-based Complementation *Problems*

Source of state explosion:

- size of $\mathcal{A}^{\complement}$ depends on the factorial of the rank bound
  - the maximum finite rank of $\mathcal{G}_w$ for $w \notin \mathcal{L}(\mathcal{A})$
  - e.g., $\{q, r, s, t\}$, $bound = 5$, $\rightsquigarrow$
    - $\{q{:}5, r{:}5, s{:}3, t{:}1\}, \{q{:}3, r{:}3, s{:}1, t{:}5\}, \{q{:}1, r{:}3, s{:}1, t{:}5\}, \ldots$
- by default, $bound = 2|Q| - 1$
  - often unnecessarily high! $\rightsquigarrow$ many redundant states generated

Rank-based Complementation *Problems*

Source of state explosion:

- size of $\mathcal{A}^\complement$ depends on the factorial of the rank bound
  - the maximum finite rank of $\mathcal{G}_w$ for $w \notin \mathcal{L}(\mathcal{A})$
  - e.g., $\{q, r, s, t\}$, $bound = 5$, $\leadsto$
    - $\{q{:}5, r{:}5, s{:}3, t{:}1\}, \{q{:}3, r{:}3, s{:}1, t{:}5\}, \{q{:}1, r{:}3, s{:}1, t{:}5\}, \ldots$
- by default, $bound = 2|Q| - 1$
  - often unnecessarily high! $\leadsto$ many redundant states generated

This talk

Get a safe rank bound for every state by:

1. considering BAs with restricted structure (elevator BA)
2. data-flow analysis for general BAs

Rank-based Complementation *Problems*

Source of state explosion:

- size of $\mathcal{A}^\complement$ depends on the factorial of the rank bound
  - the maximum finite rank of $\mathcal{G}_w$ for $w \notin \mathcal{L}(\mathcal{A})$
  - e.g., $\{q, r, s, t\}$, $bound = 5$, $\rightsquigarrow$
    - $\{q{:}5, r{:}5, s{:}3, t{:}1\}, \{q{:}3, r{:}3, s{:}1, t{:}5\}, \{q{:}1, r{:}3, s{:}1, t{:}5\}, \ldots$
- by default, $bound = 2|Q| - 1$
  - often unnecessarily high! $\rightsquigarrow$ many redundant states generated

This talk

Get a safe rank bound for every state by:

1. considering BAs with restricted structure (elevator BA)
2. data-flow analysis for general BAs

Keep the rank bounds as small as possible!

Elevator Automata

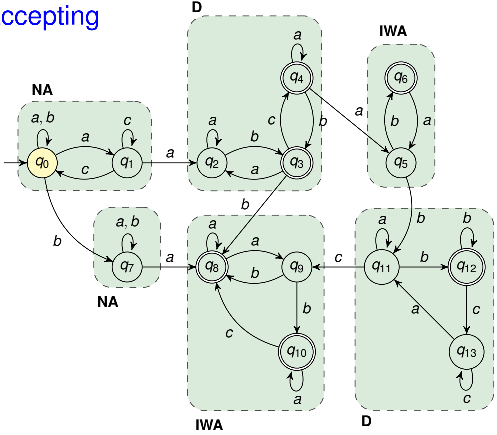
Elevator Automata

Elevator automata:

- Büchi automata with the following types of SCCs:
  - **D**: deterministic
  - **IWA**: inherently weak accepting (all cycles accepting)
  - **NA**: non-accepting

Elevator automata:

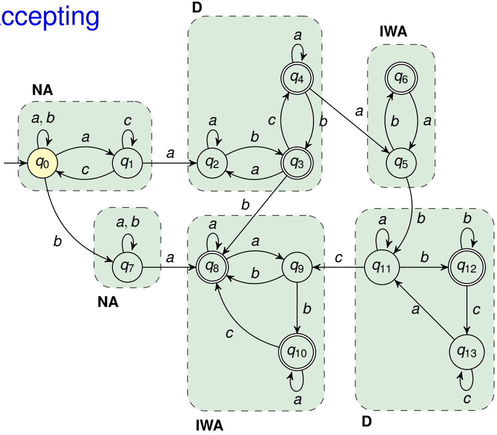
- Büchi automata with the following types of SCCs:
  - **D**: deterministic
  - **IWA**: inherently weak accepting (all cycles accepting)
  - **NA**: non-accepting



Elevator Automata

Elevator automata:

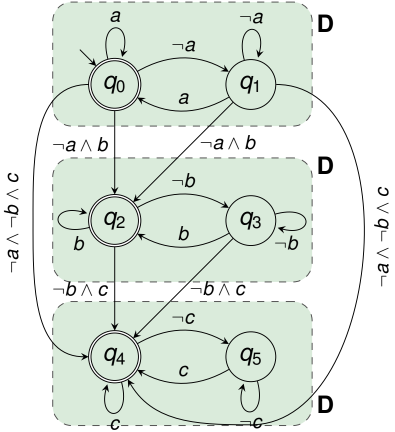
- Büchi automata with the following types of SCCs:
  - **D**: deterministic
  - **IWA**: inherently weak accepting (all cycles accepting)
  - **NA**: non-accepting



- generalization of semi-deterministic BAs (**NA** followed by **D**)

Elevator Automata

- Elevator automata often occur in practice.
  - e.g., in translation from LTL formulae (90 % of LTL benchmark)



$\mathrm{GF}(a \vee \mathrm{GF}(b \vee \mathrm{GF}c))$

Complementation of Elevator Automata

- Let us look at the condensation of $\mathcal{A}$
- *depth*$(\mathcal{A}) =$ length of longest path of $\mathcal{A}$'s condensation

Lemma

If $\mathcal{A}$ is an elevator automaton, then bound$(\mathcal{A}) \leq 2 \cdot$ depth$(\mathcal{A})$.

- for general BAs: *bound*$(\mathcal{A}) \leq 2|Q| - 1$
- new rank bound independent on $|Q| = n$

Complementation of Elevator Automata

- Let us look at the condensation of $\mathcal{A}$
- *depth*$(\mathcal{A}) =$ length of longest path of $\mathcal{A}$'s condensation

Lemma

If $\mathcal{A}$ is an elevator automaton, then bound$(\mathcal{A}) \leq 2 \cdot$ depth$(\mathcal{A})$.

- for general BAs: *bound*$(\mathcal{A}) \leq 2|Q| - 1$
- new rank bound independent on $|Q| = n$

Good, but could be better!

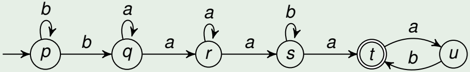
Complementation of Elevator Automata

- Why is *bound*$(\mathcal{A}) \leq 2 \cdot$ *depth*$(\mathcal{A})$ not good enough?

Complementation of Elevator Automata

- Why is $\mathit{bound}(\mathcal{A}) \leq 2 \cdot \mathit{depth}(\mathcal{A})$ not good enough?

Example



$$\mathit{bound}(\mathcal{A}) \leq 10$$

Complementation of Elevator Automata

- Why is $\mathit{bound}(\mathcal{A}) \leq 2 \cdot \mathit{depth}(\mathcal{A})$ not good enough?

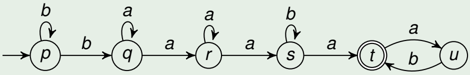
Complementation of Elevator Automata

- Why is *bound*($\mathcal{A}$) $\leq 2 \cdot$ *depth*($\mathcal{A}$) not good enough?

What to do:

- compute rank bounds for each state independently
  - can be much lower than *bound*($\mathcal{A}$) for many states!

Example



- *bound*(t) ≤ 2
- *bound*(s) $\leq 4, \ldots$

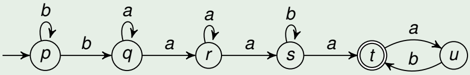
Complementation of Elevator Automata

- Why is $\mathit{bound}(\mathcal{A}) \leq 2 \cdot \mathit{depth}(\mathcal{A})$ not good enough?

What to do:

- compute rank bounds for each state independently
  - can be much lower than $\mathit{bound}(\mathcal{A})$ for many states!

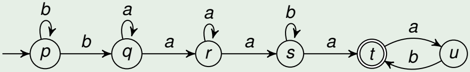
Example



- $\mathit{bound}(t) \leq 2$
- $\mathit{bound}(s) \leq 4, \ldots$

- take into account types of neighbouring SCCs

Example



- $\mathit{bound}(t) \leq 2$
- $\mathit{bound}(\{s, r, q, p\}) \leq 3, \ldots$

  - instead of changing definition, we provide algorithm

Complementation of Elevator Automata

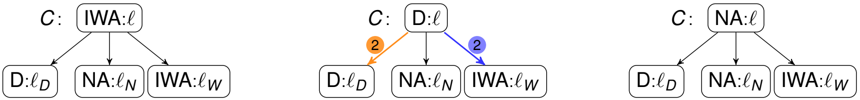
Algorithm for tighter bounds for elevator automata:

- traverse $\mathcal{A}$ back to front and apply rules to set types and ranks:
- Terminal components:
  - IWA:0 for inherently weak accepting
  - D:2 otherwise

Algorithm for tighter bounds for elevator automata:

- traverse $\mathcal{A}$ back to front and apply rules to set types and ranks:
- Terminal components:
  - IWA:0 for inherently weak accepting
  - D:2 otherwise
- Non-terminal components:

$\ell = \max\{\ell_D, \ell_N + 1, \ell_W\}$ $\quad$ $\ell = \max\{\ell_D + 2, \ell_N + 1, \ell_W + 2, 2\}$ $\quad$ $\ell = \max\{\ell_D + 1, \ell_N, \ell_W + 1\}$



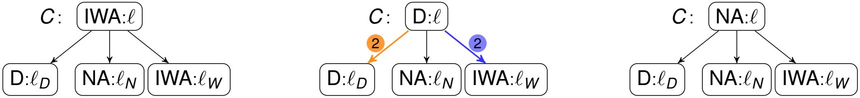
(a) *C* is **IWA** $\quad$ (b) *C* is **D** $\quad$ (c) *C* is **NA**

Complementation of Elevator Automata

Algorithm for tighter bounds for elevator automata:

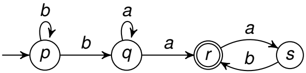
- traverse $\mathcal{A}$ back to front and apply rules to set types and ranks:
- Terminal components:
  - IWA:0 for inherently weak accepting
  - D:2 otherwise
- Non-terminal components:

$\ell = \max\{\ell_D, \ell_N + 1, \ell_W\}$ $\qquad$ $\ell = \max\{\ell_D + 2, \ell_N + 1, \ell_W + 2, 2\}$ $\qquad$ $\ell = \max\{\ell_D + 1, \ell_N, \ell_W + 1\}$



(a) C is **IWA** $\qquad$ (b) C is **D** $\qquad$ (c) C is **NA**

Example

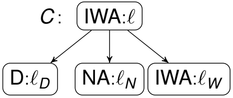


Complementation of Elevator Automata

Algorithm for tighter bounds for elevator automata:

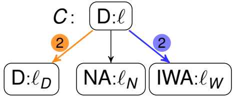
- traverse $\mathcal{A}$ back to front and apply rules to set types and ranks:
- Terminal components:
  - IWA:0 for inherently weak accepting
  - D:2 otherwise
- Non-terminal components:

$\ell = \max\{\ell_D, \ell_N + 1, \ell_W\}$



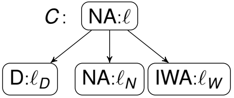
(a) *C* is **IWA**

$\ell = \max\{\ell_D + 2, \ell_N + 1, \ell_W + 2, 2\}$



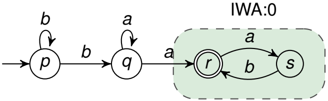
(b) *C* is **D**

$\ell = \max\{\ell_D + 1, \ell_N, \ell_W + 1\}$



(c) *C* is **NA**

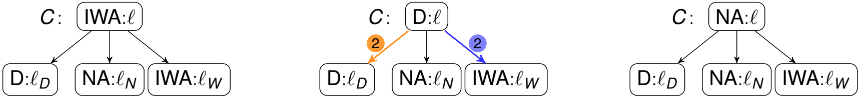
Example



Algorithm for tighter bounds for elevator automata:

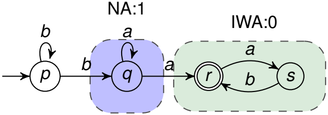
- traverse $\mathcal{A}$ back to front and apply rules to set types and ranks:
- Terminal components:
  - IWA:0 for inherently weak accepting
  - D:2 otherwise
- Non-terminal components:

$\ell = \max\{\ell_D, \ell_N + 1, \ell_W\}$ $\quad$ $\ell = \max\{\ell_D + 2, \ell_N + 1, \ell_W + 2, 2\}$ $\quad$ $\ell = \max\{\ell_D + 1, \ell_N, \ell_W + 1\}$



(a) *C* is **IWA** $\quad$ (b) *C* is **D** $\quad$ (c) *C* is **NA**

Example

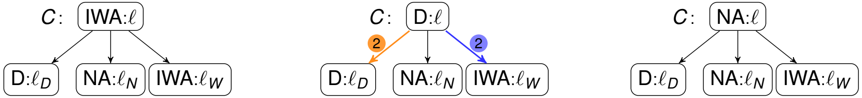


Complementation of Elevator Automata

Algorithm for tighter bounds for elevator automata:

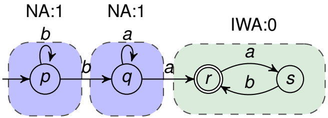
- traverse $\mathcal{A}$ back to front and apply rules to set types and ranks:
- Terminal components:
  - IWA:0 for inherently weak accepting
  - D:2 otherwise
- Non-terminal components:

$\ell = \max\{\ell_D, \ell_N + 1, \ell_W\}$ $\quad$ $\ell = \max\{\ell_D + 2, \ell_N + 1, \ell_W + 2, 2\}$ $\quad$ $\ell = \max\{\ell_D + 1, \ell_N, \ell_W + 1\}$



(a) C is **IWA** $\quad$ (b) C is **D** $\quad$ (c) C is **NA**

Example

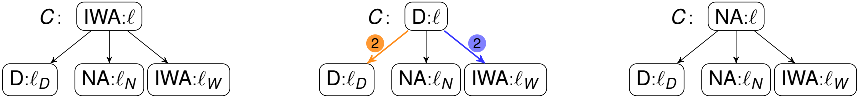


Complementation of Elevator Automata

Algorithm for tighter bounds for elevator automata:

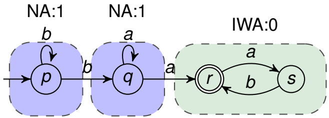
- traverse $\mathcal{A}$ back to front and apply rules to set types and ranks:
- Terminal components:
  - IWA:0 for inherently weak accepting
  - D:2 otherwise
- Non-terminal components:

$\ell = \max\{\ell_D, \ell_N + 1, \ell_W\}$ $\quad$ $\ell = \max\{\ell_D + 2, \ell_N + 1, \ell_W + 2, 2\}$ $\quad$ $\ell = \max\{\ell_D + 1, \ell_N, \ell_W + 1\}$



(a) C is **IWA** $\quad$ (b) C is **D** $\quad$ (c) C is **NA**

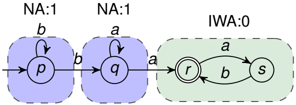
Example



max bound: 1
(lemma gives $2 \cdot 3 = 6$)

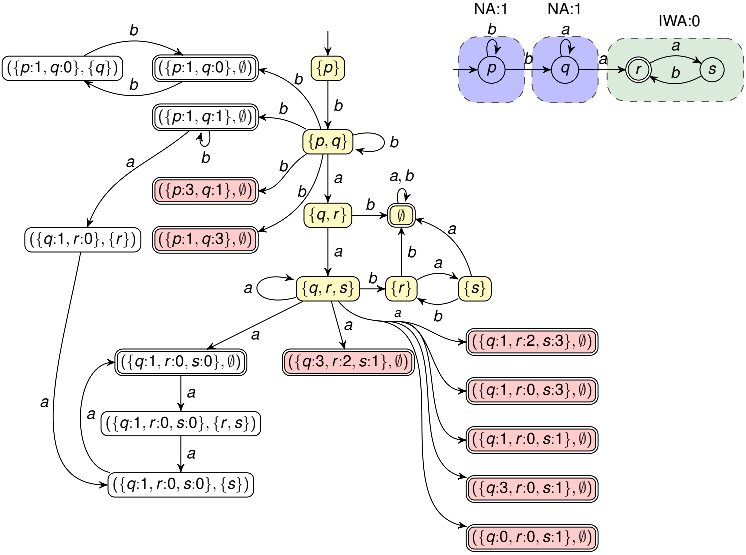
Complementation of Elevator Automata – Example

- comparison with [Schewe'09]



Complementation of Elevator Automata – Example

- comparison with [Schewe'09]



Complementation of Elevator Automata

Can we get a theoretical result not talking about number of SCCs?

Can we get a theoretical result not talking about number of SCCs?

Theorem

For elevator automaton $\mathcal{A}$, we can construct $\mathcal{A}^{\complement}$ with $\mathcal{O}(16^n)$ states.

- in general: $\mathcal{O}((0.76n)^n)$
- how to obtain $\mathcal{O}(16^n)$?

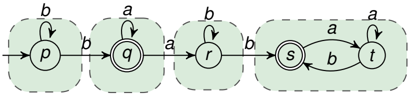
Complementation of Elevator Automata

Can we get a theoretical result not talking about number of SCCs?

Theorem

For elevator automaton $\mathcal{A}$, we can construct $\mathcal{A}^{\complement}$ with $\mathcal{O}(16^n)$ states.

- in general: $\mathcal{O}((0.76n)^n)$
- how to obtain $\mathcal{O}(16^n)$? semi-determinize!!
  - $\rightsquigarrow$ double the number of states, rank bound at most 3



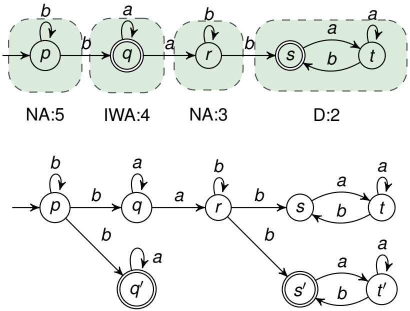
Complementation of Elevator Automata

Can we get a theoretical result not talking about number of SCCs?

Theorem

For elevator automaton $\mathcal{A}$, we can construct $\mathcal{A}^{\complement}$ with $\mathcal{O}(16^n)$ states.

- in general: $\mathcal{O}((0.76n)^n)$
- how to obtain $\mathcal{O}(16^n)$? semi-determinize!!
  - $\rightsquigarrow$ double the number of states, rank bound at most 3



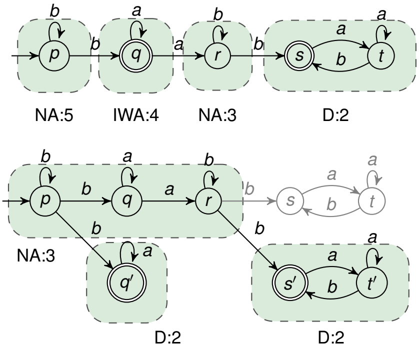
Complementation of Elevator Automata

Can we get a theoretical result not talking about number of SCCs?

Theorem

For elevator automaton $\mathcal{A}$, we can construct $\mathcal{A}^{\complement}$ with $\mathcal{O}(16^n)$ states.

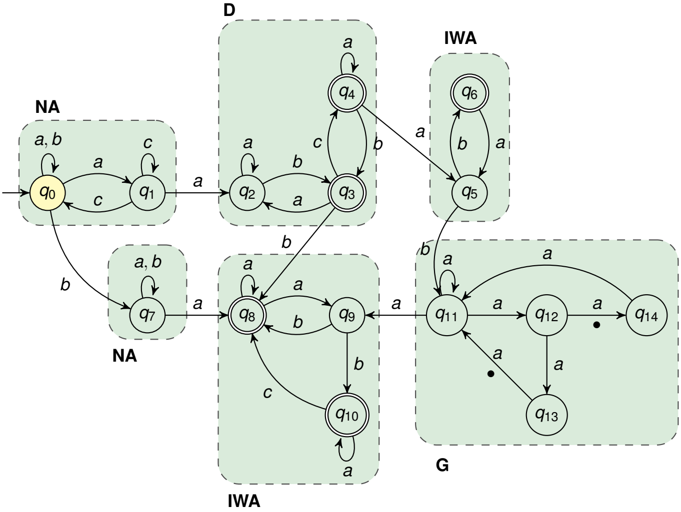
- in general: $\mathcal{O}((0.76n)^n)$
- how to obtain $\mathcal{O}(16^n)$? semi-determinize!!
  - $\rightsquigarrow$ double the number of states, rank bound at most 3



Complementation of Non-elevator BAs

Going beyond elevator automata

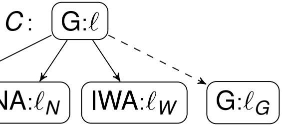
Going beyond elevator automata



Going beyond elevator automata

- the technique generalizes to non-elevator automata:
  - **G**: general SCC
- we can generalize the rules:

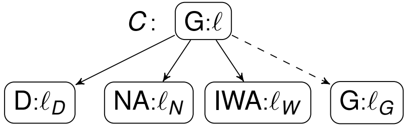
$$\ell = \max\{\ell_D, \ell_N + 1, \ell_W, \ell_G\} + 2|C \setminus Acc|$$



Going beyond elevator automata

- the technique generalizes to non-elevator automata:
  - **G**: general SCC
- we can generalize the rules:

$$\ell = \max\{\ell_D, \ell_N + 1, \ell_W, \ell_G\} + 2|C \setminus Acc|$$



- Can we improve over the $+2|C \setminus Acc|$?

Complementation of Non-elevator BAs

- Can we improve over the $+2|C \setminus Acc|$?

Complementation of Non-elevator BAs

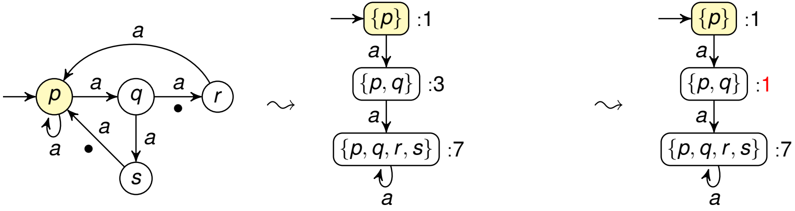
- Can we improve over the $+2|C \setminus Acc|$?
- Often, rank bounds of states within an SCC depend on each other.

Complementation of Non-elevator BAs

- Can we improve over the $+2|C \setminus Acc|$?
- Often, rank bounds of states within an SCC depend on each other.
- $\rightsquigarrow$ data flow analysis!
  - propagates rank bounds
  - outer macrostate analysis
  - inner macrostate analysis

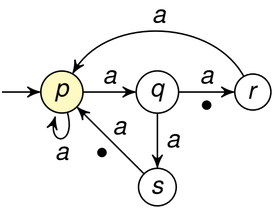
Data Flow Analysis — Outer Macrostate Analysis

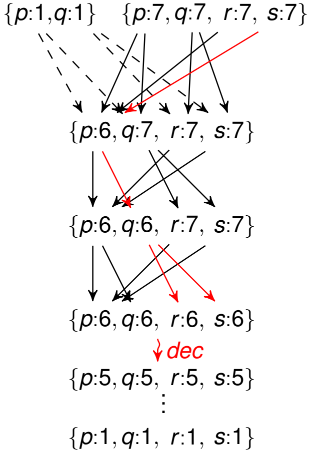
- Based on sizes of macrostates
- Bound for the smallest macrostate in every cycle
- Forward rank propagation



Data Flow Analysis — Inner Macrostate Analysis

- Based on ranks assigned to all states in a macrostate





Experiments

Experimental Evaluation

- Random automata from [Tsai,Fogarty,Vardi,Tsay'11]
  - alphabet of 2 symbols
  - starting with 15 states
  - reduced using SPOT, RABIT
  - removed semi-deterministic, inherently weak, unambiguous, empty
  - 2592 hard automata
  - Timeout: 5 min

Experimental Evaluation

- Random automata from [Tsai,Fogarty,Vardi,Tsay'11]
  - alphabet of 2 symbols
  - starting with 15 states
  - reduced using SPOT, RABIT
  - removed semi-deterministic, inherently weak, unambiguous, empty
  - 2592 hard automata
  - Timeout: 5 min
- LTL automata
  - larger alphabets (up to 128 symbols)
  - from LTL formulae (from literature and randomly generated)
  - 414 hard automata
  - Timeout: 5 min

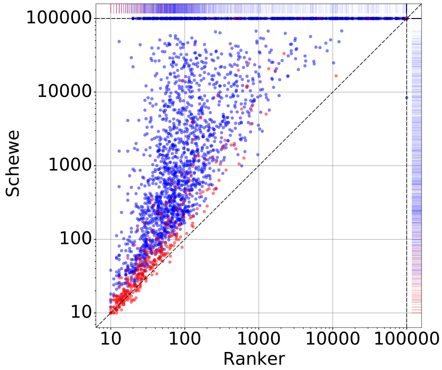
Experimental Evaluation

- Random automata from [Tsai,Fogarty,Vardi,Tsay'11]
  - alphabet of 2 symbols
  - starting with 15 states
  - reduced using SPOT, RABIT
  - removed semi-deterministic, inherently weak, unambiguous, empty
  - 2592 hard automata
  - Timeout: 5 min
- LTL automata
  - larger alphabets (up to 128 symbols)
  - from LTL formulae (from literature and randomly generated)
  - 414 hard automata
  - Timeout: 5 min
- Total: 3006 state-based BAs, 458 of them elevator automata

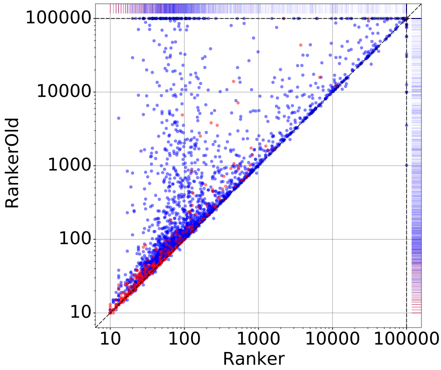
Experimental Evaluation

- Implemented in C++ within RANKER
- Compared with:
  - GOAL⚽ (SCHEWE, SAFRA, PITERMAN, FRIBOURG)
  - SPOT
  - LTL2DSTAR
  - SEMINATOR 2
  - ROLL

Experimental Evaluation – States *rank-based*



(a) RANKER vs SCHEWE

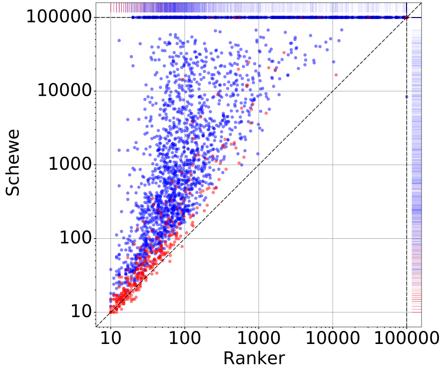


(b) RANKER vs $\text{RANKER}_{\text{OLD}}$

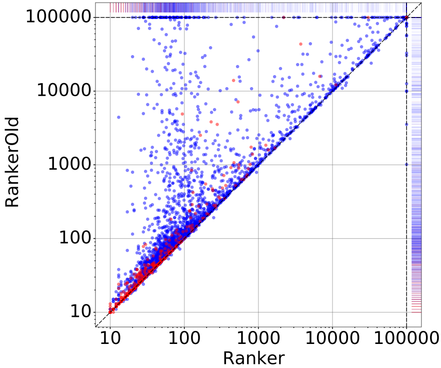
- SCHEWE: [Schewe'09]
- $\text{RANKER}_{\text{OLD}}$: [Havlena,L.'21]

- **blue**: random
- **red**: LTL
- no post-processing

Experimental Evaluation – States *rank-based*



(a) RANKER vs SCHEWE

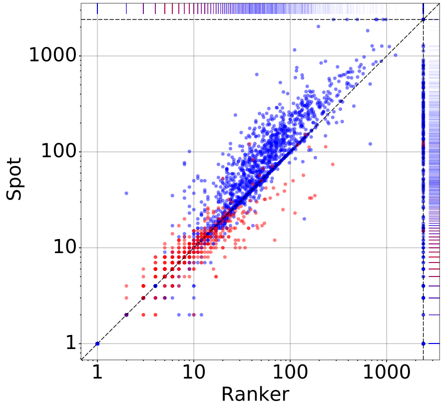


(b) RANKER vs $\text{RANKER}_{\text{OLD}}$

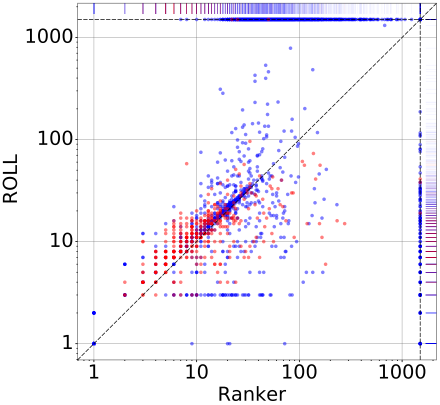
method	mean	median	wins	losses	timeouts
RANKER	**3812** (4452 : 207)	**79** (93 : 26)			**279** (276 : 3)
$\text{RANKER}_{\text{OLD}}$	7398 (8688 : 358)	141 (197 : 29)	2190 (2011 : 179)	111 (107 : 4)	365 (360 : 5)
SCHEWE	4550 (5495 : 665)	439 (774 : 35)	2640 (2315 : 325)	55 (1 : 54)	937 (928 : 9)

- all (random : LTL)

Experimental Evaluation – States *not rank-based*



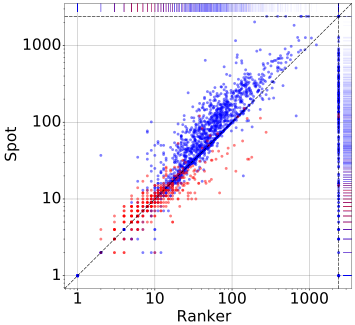
(a) RANKER vs SPOT



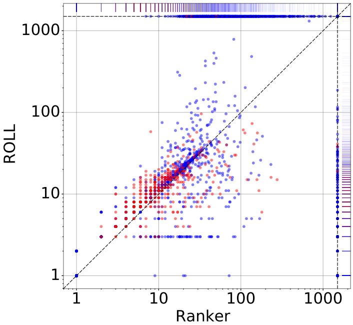
(b) RANKER vs ROLL

- SPOT: determinisation-based [Duret-Lutz et al.'16]
- ROLL: learning-based [Li et al.'19]

- **blue**: random
- **red**: LTL
- post-processing: SPOT



(a) RANKER vs SPOT



(b) RANKER vs ROLL

method	mean	median	wins	losses	timeouts
RANKER	**47** (52 : 18)	**22** (27 : 10)			279 (276 : 3)
PITERMAN	**73** (82 : 22)	28 (34 : 14)	1435 (1124 : 311)	416 (360 : 56)	**14** (12 : 2)
SAFRA	83 (91 : 30)	29 (35 : 17)	1562 (1211 : 351)	387 (350 : 37)	172 (158 : 14)
SPOT	75 (85 : 15)	24 (32 : 10)	1087 (936 : 151)	683 (501 : 182)	**13** (13 : 0)
FRIBOURG	91 (104 : 13)	23 (31 : 9)	1120 (1055 : 65)	601 (376 : 225)	**81** (80 : 1)
LTL2DSTAR	**73** (82 : 21)	28 (34 : 13)	1465 (1195 : 270)	465 (383 : 82)	136 (130 : 6)
SEMINATOR 2	79 (91 : 15)	**21** (29 : 10)	1266 (1131 : 135)	571 (367 : 204)	363 (362 : 1)
ROLL	**18** (19 : 14)	**10** (9 : 11)	2116 (1858 : 258)	569 (443 : 126)	1109 (1106 : 3)

- all (random : LTL)

Future Work

- Generalization to complementation of TELA
  - transition-based Emerson-Lei automata
- Exploit the elevator structure even more
- Language inclusion checking

Conclusion

- **Elevator automata**
  - BAs with deterministic, inherently weak, and non-accepting SCCs
  - occur often in practice
  - the structure can be exploited

- in **rank-based complementation**
  - allow tighter bounds of states' ranks $\rightsquigarrow$ smaller $\mathcal{A}^{\complement}$
  - can be generalized to general SCCs
  - rank bounds can be refined by data-flow analysis

Conclusion

- **Elevator automata**
  - BAs with deterministic, inherently weak, and non-accepting SCCs
  - occur often in practice
  - the structure can be exploited

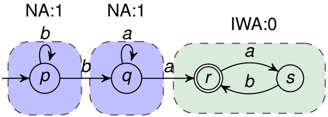
- in **rank-based complementation**
  - allow tighter bounds of states' ranks $\rightsquigarrow$ smaller $\mathcal{A}^{\complement}$
  - can be generalized to general SCCs
  - rank bounds can be refined by data-flow analysis

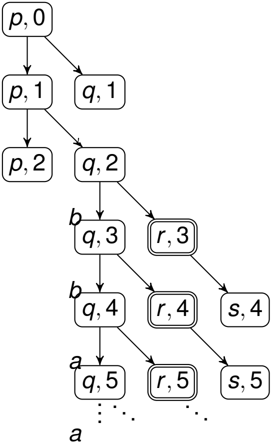
THANK YOU!

Experimental Evaluation – Time

method	mean runtime [s]				median runtime [s]				timeouts			
RANKER	7.83	(8.99	:	1.30)	0.51	(0.84	:	0.04)	279	(276	:	3)
$\text{RANKER}_{\text{OLD}}$	9.37	(10.73	:	1.99)	0.61	(1.04	:	0.04)	365	(360	:	5)
SCHEWE	21.05	(24.28	:	7.80)	6.57	(7.39	:	5.21)	937	(928	:	9)
RANKER	7.83	(8.99	:	1.30)	0.51	(0.84	:	0.04)	279	(276	:	3)
PITERMAN	7.29	(7.39	:	6.65)	5.99	(6.04	:	5.62)	14	(12	:	2)
SAFRA	14.11	(15.05	:	8.37)	6.71	(6.92	:	5.79)	172	(158	:	14)
SPOT	0.86	(0.99	:	0.06)	0.02	(0.02	:	0.02)	13	(13	:	0)
FRIBOURG	17.79	(19.53	:	7.22)	9.25	(10.15	:	5.48)	81	(80	:	1)
LTL2DSTAR	3.31	(3.84	:	0.11)	0.04	(0.05	:	0.02)	136	(130	:	6)
SEMINATOR 2	9.51	(11.25	:	0.08)	0.22	(0.39	:	0.02)	363	(362	:	1)
ROLL	31.23	(37.85	:	7.28)	8.19	(12.23	:	2.74)	1109	(1106	:	3)

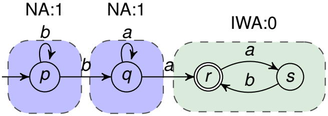
- $bba^\omega \notin \mathcal{L}(\mathcal{A})$

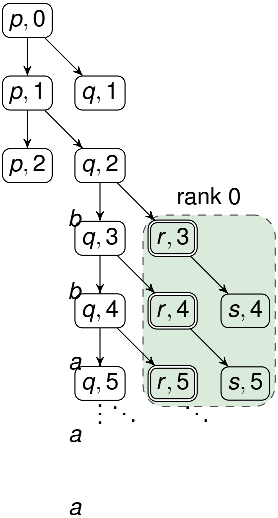




Elevator Automata Complementation – Run DAGs

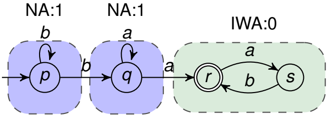
- $bba^\omega \notin \mathcal{L}(\mathcal{A})$

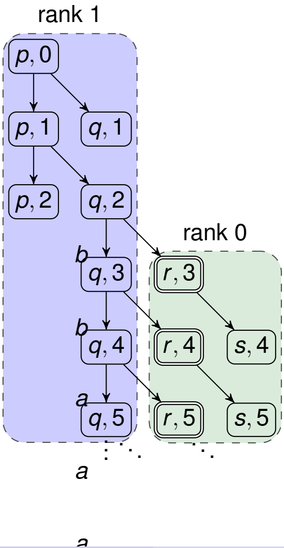




Elevator Automata Complementation – Run DAGs

- $bba^\omega \notin \mathcal{L}(\mathcal{A})$





Rank-based Complementation

- Nondeterministically guesses run DAG ranks [Schewe'09]

Rank-based Complementation

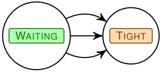
- Nondeterministically guesses run DAG ranks [Schewe'09]
- Macrostates (S, O, f, i); accepting macrostates $(\cdot, \emptyset, \cdot, \cdot)$ (omit i)
  - S tracks all runs of $\mathcal{A}$ (determinization of NFAs)
  - O tracks all runs with an even rank (since a breakpoint with $O = \emptyset$)
    - to accept a word $\rightsquigarrow$ decrease ranks of the runs from O
  - f guesses ranks of a level in a run DAG
    - tight rankings: (i) odd max rank r (ii) cover ranks $\{1, 3, \ldots, r\}$

Rank-based Complementation

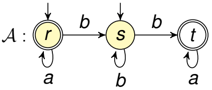
- Nondeterministically guesses run DAG ranks [Schewe'09]
- Macrostates (S, O, f, i); accepting macrostates $(\cdot, \emptyset, \cdot, \cdot)$ (omit i)
  - S tracks all runs of $\mathcal{A}$ (determinization of NFAs)
  - O tracks all runs with an even rank (since a breakpoint with $O = \emptyset$)
    - to accept a word $\rightsquigarrow$ decrease ranks of the runs from O
  - f guesses ranks of a level in a run DAG
    - tight rankings: (i) odd max rank r (ii) cover ranks $\{1, 3, \ldots, r\}$
- Transition function $\boxed{(S, O, f)} \xrightarrow{a} \boxed{(S', O', f')}$
  - S'-part: subset construction; $S' = \delta(S, a)$
  - O'-part: keeps successors of O with even ranks (or a new sample if $O = \emptyset$)
  - f': nonincreasing tight ranking wrt δ (with even accepting states)

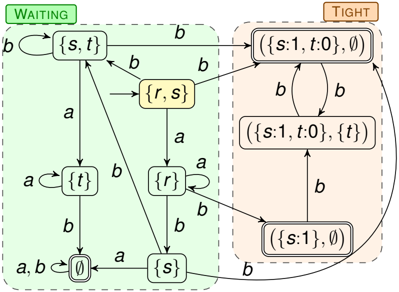
Rank-based Complementation

- Nondeterministically guesses run DAG ranks [Schewe'09]
- Macrostates (S, O, f, i); accepting macrostates $(\cdot, \emptyset, \cdot, \cdot)$ (omit i)
  - S tracks all runs of $\mathcal{A}$ (determinization of NFAs)
  - O tracks all runs with an even rank (since a breakpoint with $O = \emptyset$)
    - to accept a word $\rightsquigarrow$ decrease ranks of the runs from O
  - f guesses ranks of a level in a run DAG
    - tight rankings: (i) odd max rank r (ii) cover ranks $\{1, 3, \ldots, r\}$
- Transition function $(S, O, f) \xrightarrow{a} (S', O', f')$
  - S'-part: subset construction; $S' = \delta(S, a)$
  - O'-part: keeps successors of O with even ranks (or a new sample if $O = \emptyset$)
  - f': nonincreasing tight ranking wrt δ (with even accepting states)
- Waiting and Tight part
  - in Waiting guess the point from which all successor rankings are tight (only S-part)
  - in Tight track tight rankings

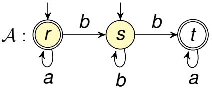


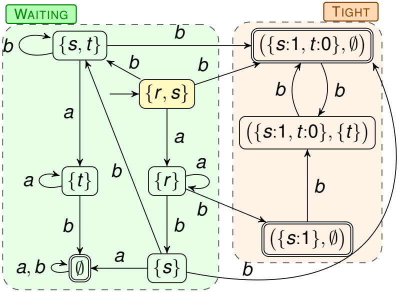
Rank-based Complementation *Example*





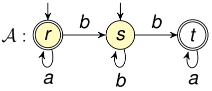
Rank-based Complementation *Example*

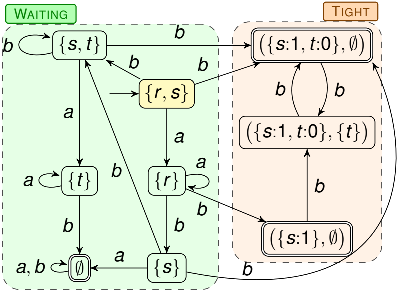




- $(\{s{:}1, t{:}0\}, \emptyset) \xrightarrow{b} (S', O', f')$
  - $S' = \delta(\{s,t\}, b) = \{s,t\}$
  - $f'(s) \leq f(s)$, $f'(t) \leq f(s)$,
    $f'(t)$ is even $\Longrightarrow \{s{:}1, t{:}0\}$
  - $O' = \{t\}$ $\quad (O' = S' \cap even(f'))$
  - $(\{s{:}1, t{:}0\}, \{t\})$

Rank-based Complementation *Example*





- $(\{s{:}1, t{:}0\}, \emptyset) \xrightarrow{b} (S', O', f')$
  - $S' = \delta(\{s, t\}, b) = \{s, t\}$
  - $f'(s) \leq f(s)$, $f'(t) \leq f(s)$, $f'(t)$ is even $\Longrightarrow \{s{:}1, t{:}0\}$
  - $O' = \{t\}$ $\quad (O' = S' \cap even(f'))$
  - $(\{s{:}1, t{:}0\}, \{t\})$
- $(\{s{:}1, t{:}0\}, \{t\}) \xrightarrow{b} (S', O', f')$
  - S', f' similar to the previous case
  - $O' = \emptyset$ $\quad (O' = \delta(\{t\}, b) \cap even(f'))$
  - $(\{s{:}1, t{:}0\}, \emptyset)$