Next Generation of Rank-Based Algorithms for Omega Automata

Barbora Šmahlíková
Vojtěch Havlena Ondřej Lengál

Brno University of Technology, Czech Republic

AVM'22

Büchi Automata (BA)

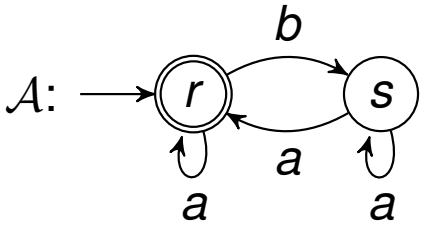
- Automata over infinite words
- $\mathcal{A} = (Q, \Sigma, \delta, I, F)$
  - Q is a finite set of states
  - Σ is an alphabet
  - $\delta : Q \times \Sigma \to 2^Q$ is a transition function
  - $I \subseteq Q$ is a set of initial states
  - $F \subseteq Q$ is a set of accepting states

Büchi Automata (BA)

- Automata over infinite words
- $\mathcal{A} = (Q, \Sigma, \delta, I, F)$
  - Q is a finite set of states
  - Σ is an alphabet
  - $\delta : Q \times \Sigma \to 2^Q$ is a transition function
  - $I \subseteq Q$ is a set of initial states
  - $F \subseteq Q$ is a set of accepting states
- Accept by going through some accepting state infinitely often

Büchi Automata (BA)

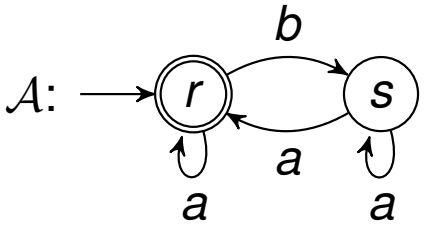
- Automata over infinite words
- $\mathcal{A} = (Q, \Sigma, \delta, I, F)$
  - Q is a finite set of states
  - Σ is an alphabet
  - $\delta : Q \times \Sigma \to 2^Q$ is a transition function
  - $I \subseteq Q$ is a set of initial states
  - $F \subseteq Q$ is a set of accepting states
- Accept by going through some accepting state infinitely often



Büchi Automata (BA)

- Automata over infinite words
- $\mathcal{A} = (Q, \Sigma, \delta, I, F)$
  - Q is a finite set of states
  - Σ is an alphabet
  - $\delta : Q \times \Sigma \to 2^Q$ is a transition function
  - $I \subseteq Q$ is a set of initial states
  - $F \subseteq Q$ is a set of accepting states

- Accept by going through some accepting state infinitely often

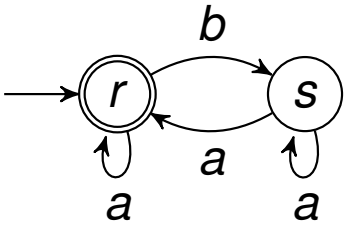


- $r \xrightarrow{a} r \xrightarrow{b} s \xrightarrow{a} r \xrightarrow{b} s \xrightarrow{a} r \xrightarrow{b} s \xrightarrow{a} \cdots$
- $(ab)^\omega \in \mathcal{L}(\mathcal{A})$

Büchi Automata (BA)

- Automata over infinite words
- $\mathcal{A} = (Q, \Sigma, \delta, I, F)$
  - Q is a finite set of states
  - Σ is an alphabet
  - $\delta : Q \times \Sigma \to 2^Q$ is a transition function
  - $I \subseteq Q$ is a set of initial states
  - $F \subseteq Q$ is a set of accepting states
- Accept by going through some accepting state infinitely often

$\mathcal{A}$:



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

- Define the class of ω-regular languages

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

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

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

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

- Termination analysis of programs: Ultimate Automizer

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

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

- Termination analysis of programs: Ultimate Automizer
- 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:

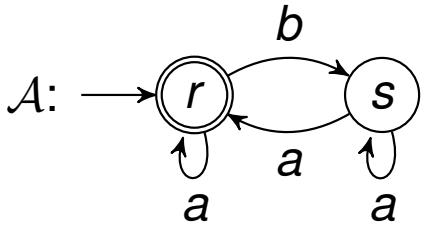
- Model checking of linear-time properties

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

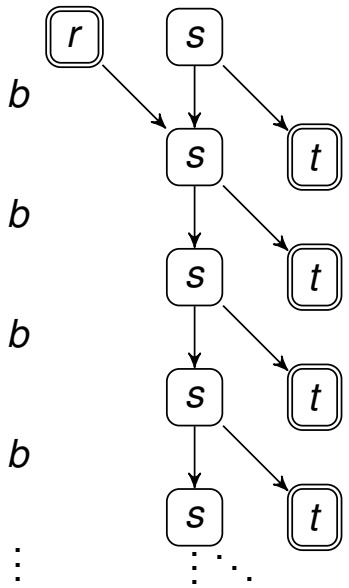
- Termination analysis of programs: Ultimate Automizer
- Decision procedures: implements negation
  - S1S: MSO over $(\omega, 0, +1)$
  - QPTL: quantified propositional temporal logic
  - FO over Sturmian words
- Basic operation for inclusion/equivalence checking

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

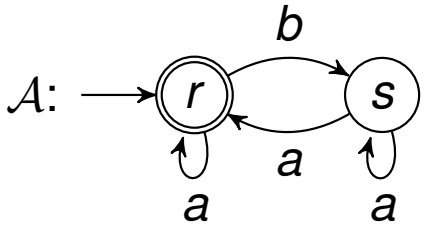


run DAG for $b^\omega \notin \mathcal{L}(\mathcal{A})$



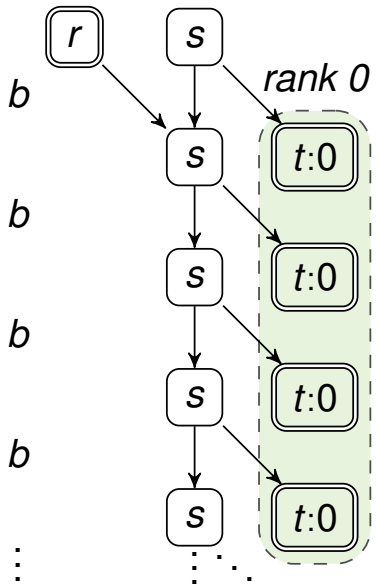
Rank-based Complementation

- 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



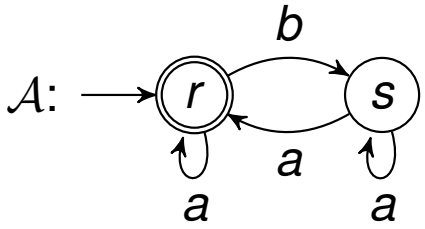
- Ranking procedure (start with $i = 0$)
  1. assign rank i to vertices with finitely many successors and remove them from $\mathcal{G}_w$

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



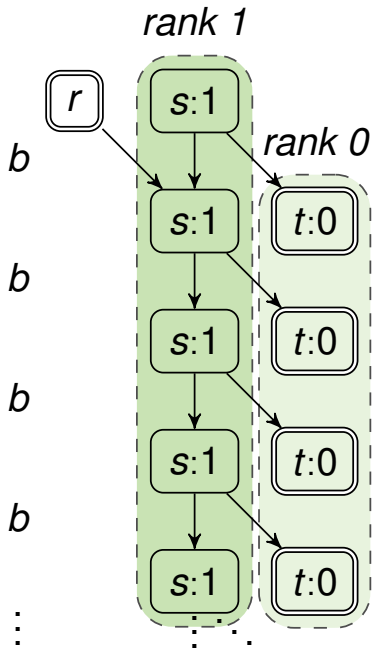
Rank-based Complementation

- 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



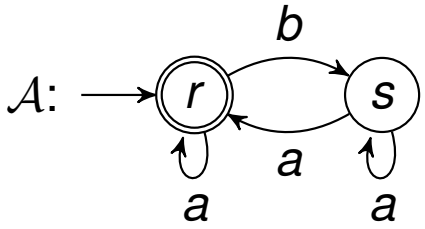
- Ranking procedure (start with $i = 0$)
  1. assign rank i to vertices with finitely many successors and remove them from $\mathcal{G}_w$
  2. assign rank $i + 1$ to vertices that cannot reach *Acc* and remove them from $\mathcal{G}_w$

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



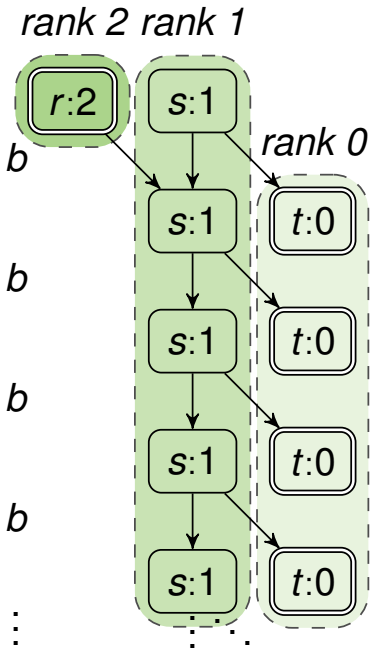
Rank-based Complementation

- 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



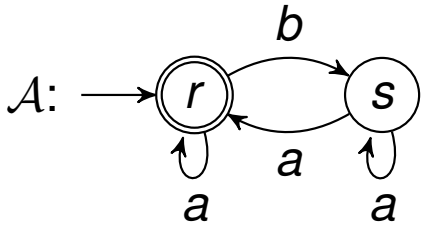
- Ranking procedure (start with $i = 0$)
  1. assign rank i to vertices with finitely many successors and remove them from $\mathcal{G}_w$
  2. assign rank $i + 1$ to vertices that cannot reach *Acc* and remove them from $\mathcal{G}_w$
  3. $i := i + 2$;
     repeat until $\mathcal{G}_w = \emptyset$

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



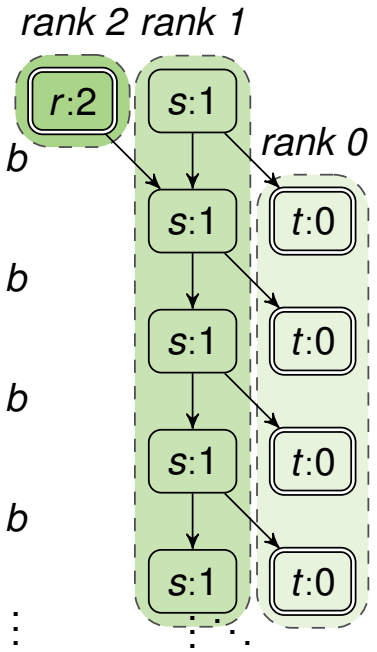
Rank-based Complementation

- 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

- 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 $\{1, 3, \ldots, r\}$

Source of state explosion:

- Complement size 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

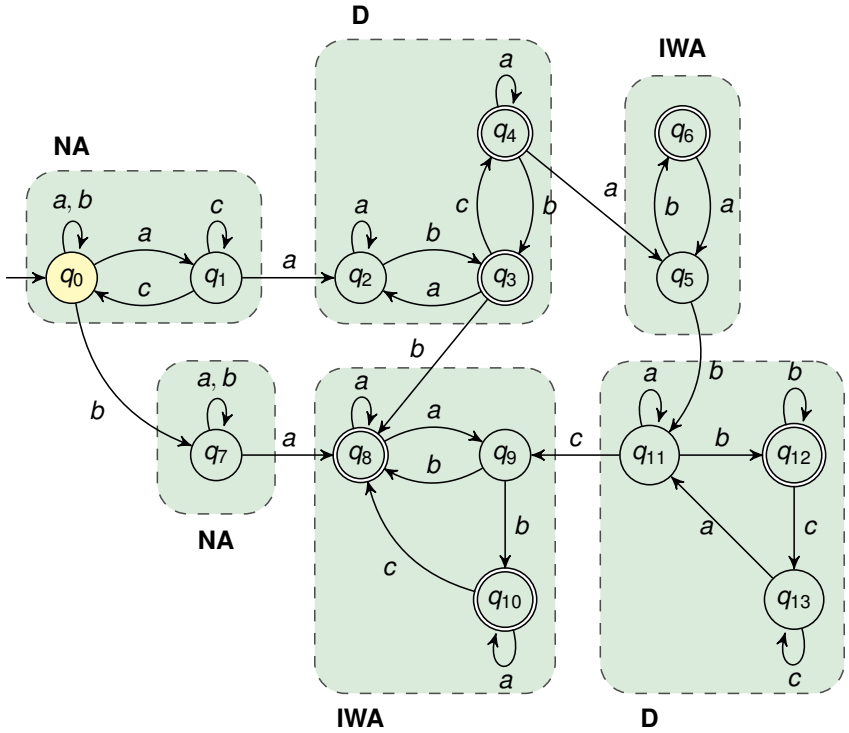
- Nondeterministically guesses run DAG ranks [Schewe'09]
- Macrostates (S, O, f, i); accepting macrostates $(\cdot, \emptyset, \cdot, \cdot)$ (omit i)
  - S tracks all runs of $\mathcal{A}$ (determinization of NFAs)
  - O tracks all runs with an even rank (since a breakpoint with $O = \emptyset$)
    - 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 $\{1, 3, \ldots, r\}$

Source of state explosion:

- Complement size 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

Elevator Automata[1]

- Subclass of Büchi Automata
- Often occur in practice (BAs from LTL formulae)
- Types of SCCs:
  - **N**: nonaccepting
  - **D**: deterministic
  - **IWA**: inherently weak accepting (every cycle contains an accepting state)



[1] Havlena, Lengál, and Šmahlíková. "Sky Is Not the Limit: Tighter Rank Bounds for Elevator Automata in Büchi Automata Complementation". In: *TACAS'22*.

Complementation of Elevator Automata

Algorithm for tighter bounds for elevator automata:

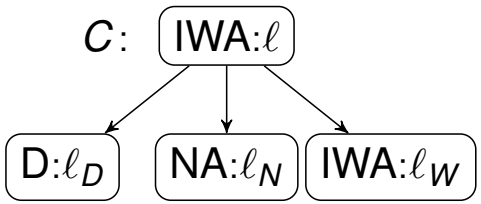
- traverse $\mathcal{A}$ back to front and apply rules to set types and ranks:
- Terminal components:
  - IWA:0 for inherently weak accepting
  - D:2 otherwise

Complementation of Elevator Automata

Algorithm for tighter bounds for elevator automata:

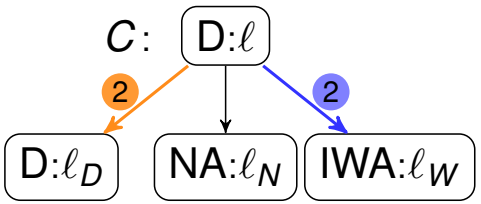
- traverse $\mathcal{A}$ back to front and apply rules to set types and ranks:
- Terminal components:
  - IWA:0 for inherently weak accepting
  - D:2 otherwise
- Non-terminal components:

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



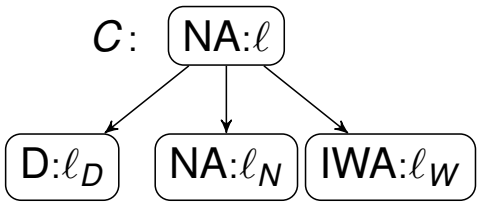
(a) C is **IWA**

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



(b) C is **D**

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



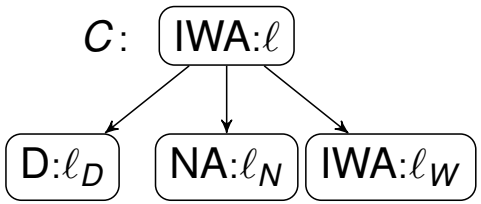
(c) C is **NA**

Complementation of Elevator Automata

Algorithm for tighter bounds for elevator automata:

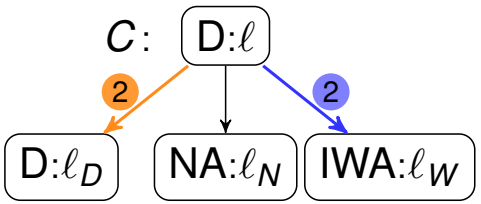
- traverse $\mathcal{A}$ back to front and apply rules to set types and ranks:
- Terminal components:
  - IWA:0 for inherently weak accepting
  - D:2 otherwise
- Non-terminal components:

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



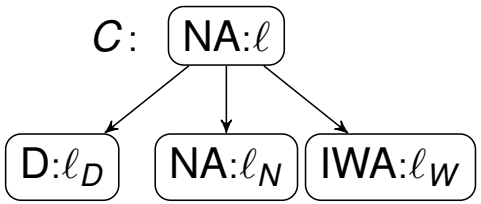
(a) C is **IWA**

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



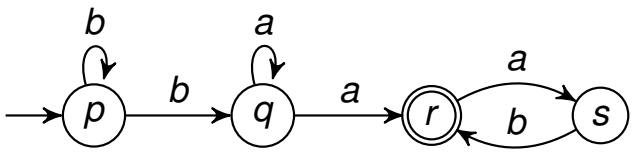
(b) C is **D**

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



(c) C is **NA**

Example

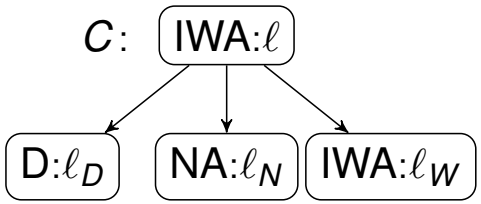


Complementation of Elevator Automata

Algorithm for tighter bounds for elevator automata:

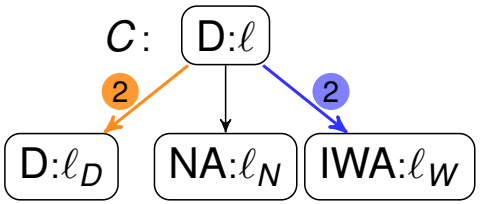
- traverse $\mathcal{A}$ back to front and apply rules to set types and ranks:
- Terminal components:
  - IWA:0 for inherently weak accepting
  - D:2 otherwise
- Non-terminal components:

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



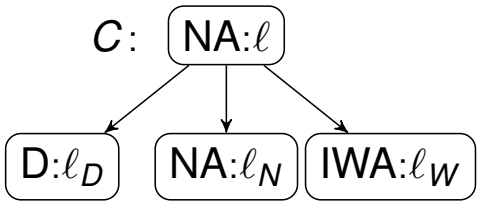
(a) C is **IWA**

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



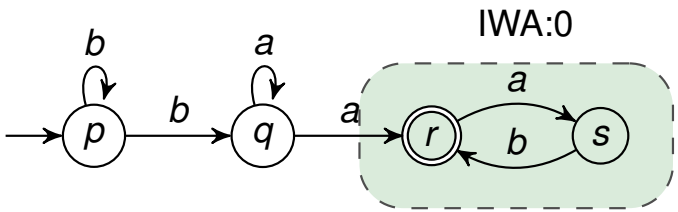
(b) C is **D**

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



(c) C is **NA**

Example

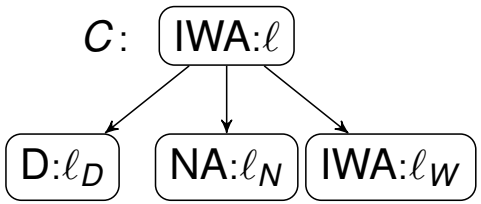


Complementation of Elevator Automata

Algorithm for tighter bounds for elevator automata:

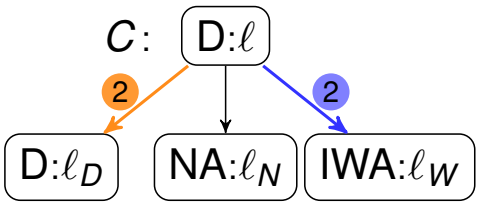
- traverse $\mathcal{A}$ back to front and apply rules to set types and ranks:
- Terminal components:
  - IWA:0 for inherently weak accepting
  - D:2 otherwise
- Non-terminal components:

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



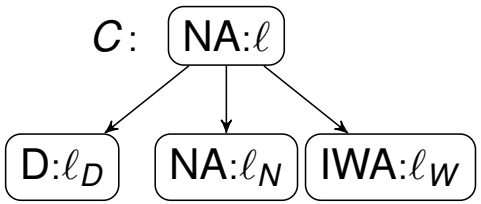
(a) C is **IWA**

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



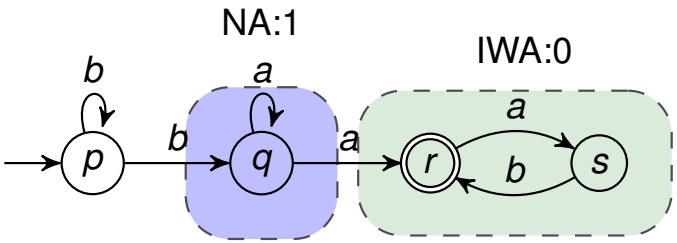
(b) C is **D**

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



(c) C is **NA**

Example

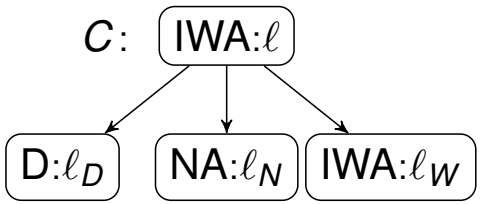


Complementation of Elevator Automata

Algorithm for tighter bounds for elevator automata:

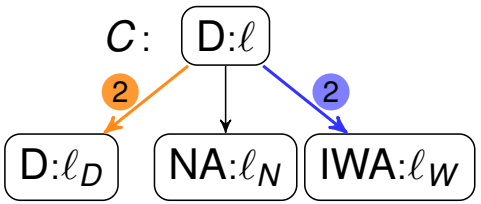
- traverse $\mathcal{A}$ back to front and apply rules to set types and ranks:
- Terminal components:
  - IWA:0 for inherently weak accepting
  - D:2 otherwise
- Non-terminal components:

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



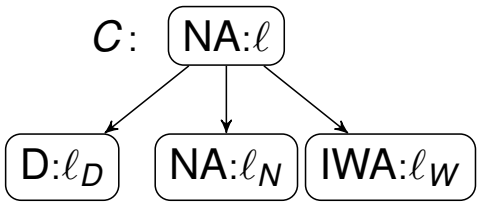
(a) C is **IWA**

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



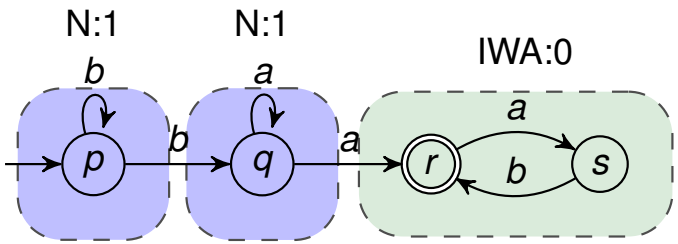
(b) C is **D**

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



(c) C is **NA**

Example

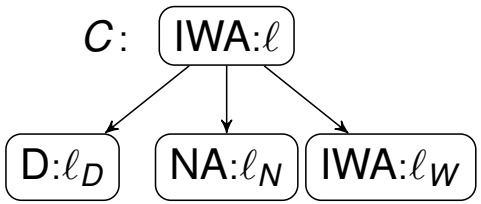


Complementation of Elevator Automata

Algorithm for tighter bounds for elevator automata:

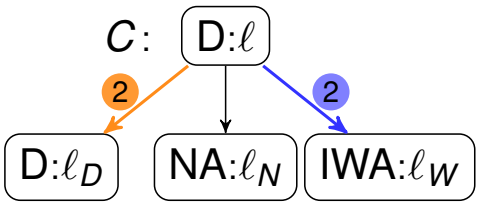
- traverse $\mathcal{A}$ back to front and apply rules to set types and ranks:
- Terminal components:
  - IWA:0 for inherently weak accepting
  - D:2 otherwise
- Non-terminal components:

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



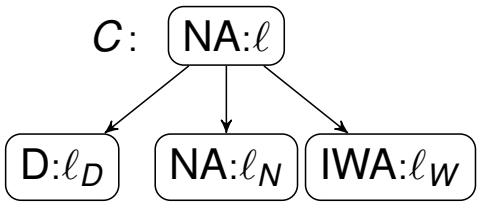
(a) C is **IWA**

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



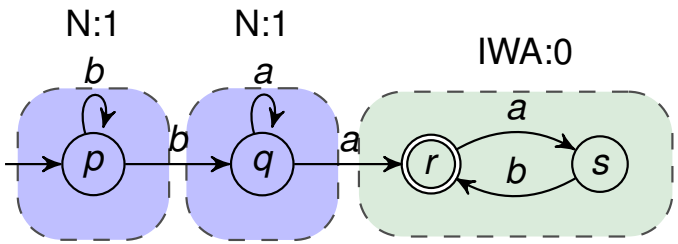
(b) C is **D**

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



(c) C is **NA**

Example

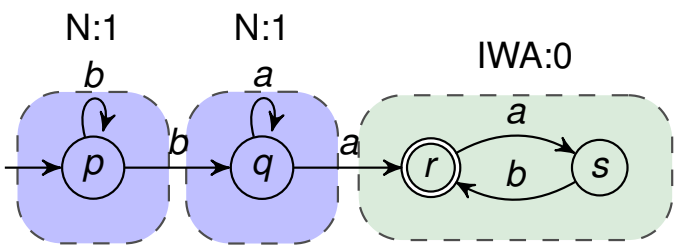


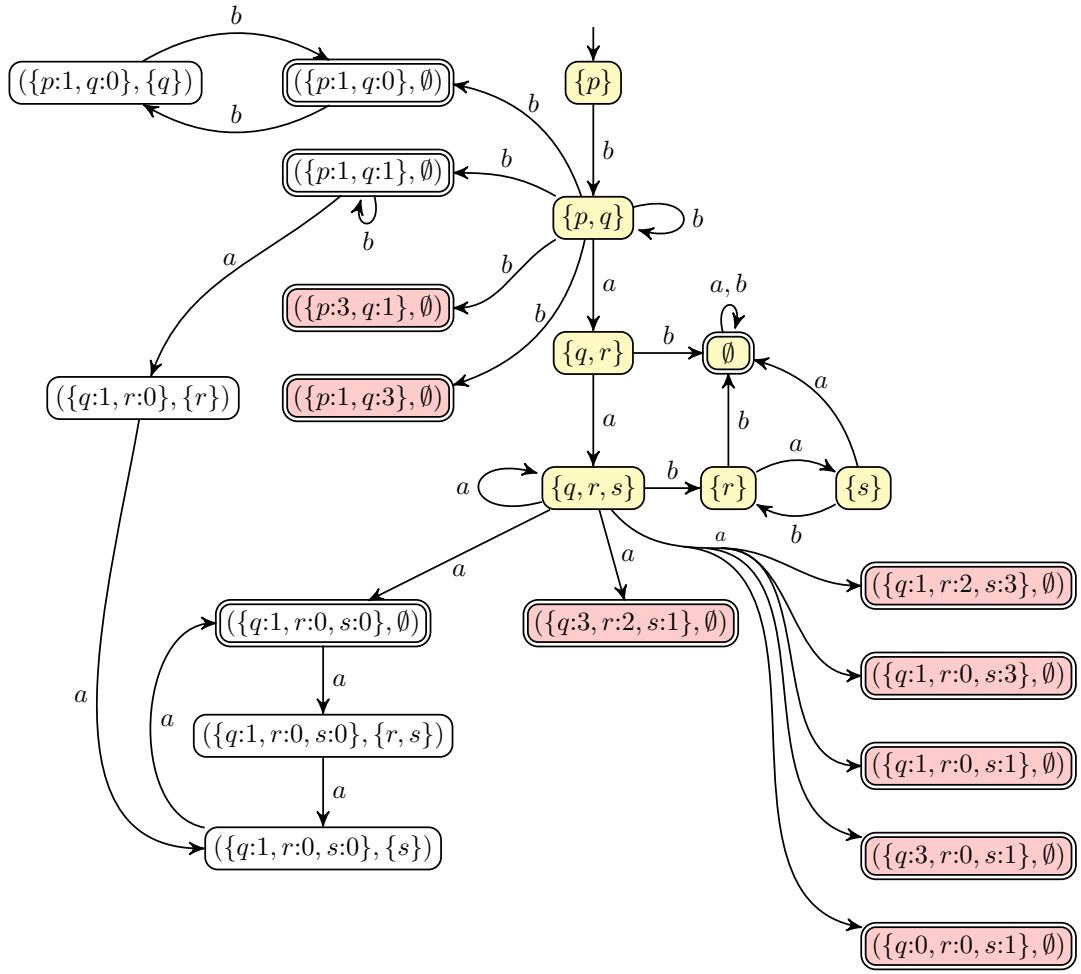
max bound: 1
(lemma gives $2 \cdot 3 = 6$)

Complementation of Elevator Automata

Example:

- Red states are not generated





Complementation of Elevator Automata

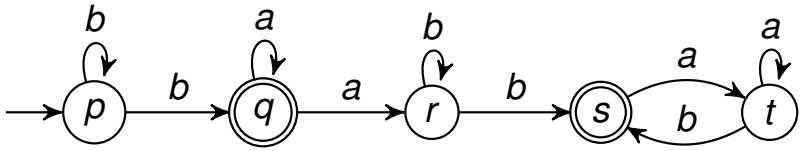
Deelevation:

- Decreases the rank bound to 3
- At most $2|Q|$ states of the input automaton

Complementation of Elevator Automata

Deelevation:

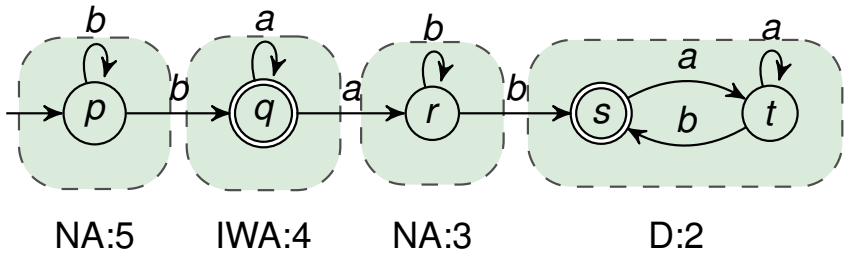
- Decreases the rank bound to 3
- At most $2|Q|$ states of the input automaton



Complementation of Elevator Automata

Deelevation:

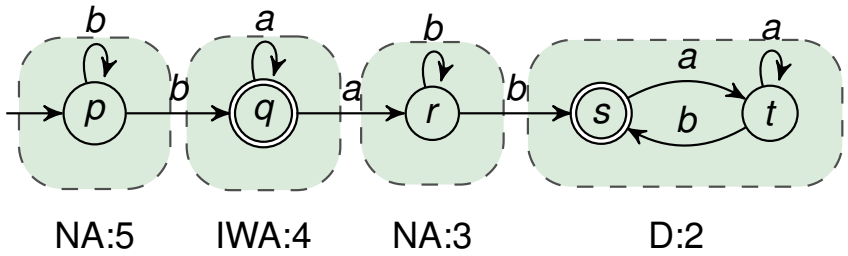
- Decreases the rank bound to 3
- At most $2|Q|$ states of the input automaton

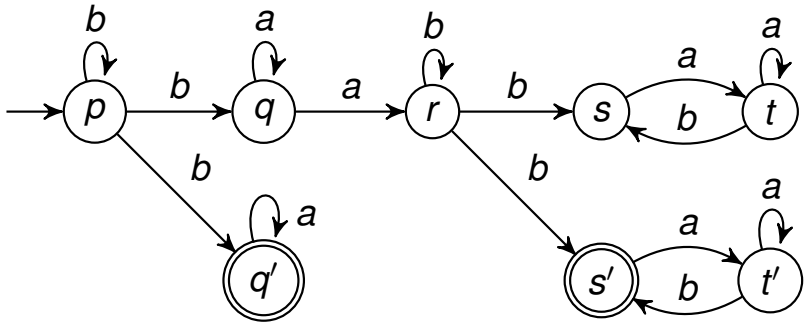


Complementation of Elevator Automata

Deelevation:

- Decreases the rank bound to 3
- At most $2|Q|$ states of the input automaton

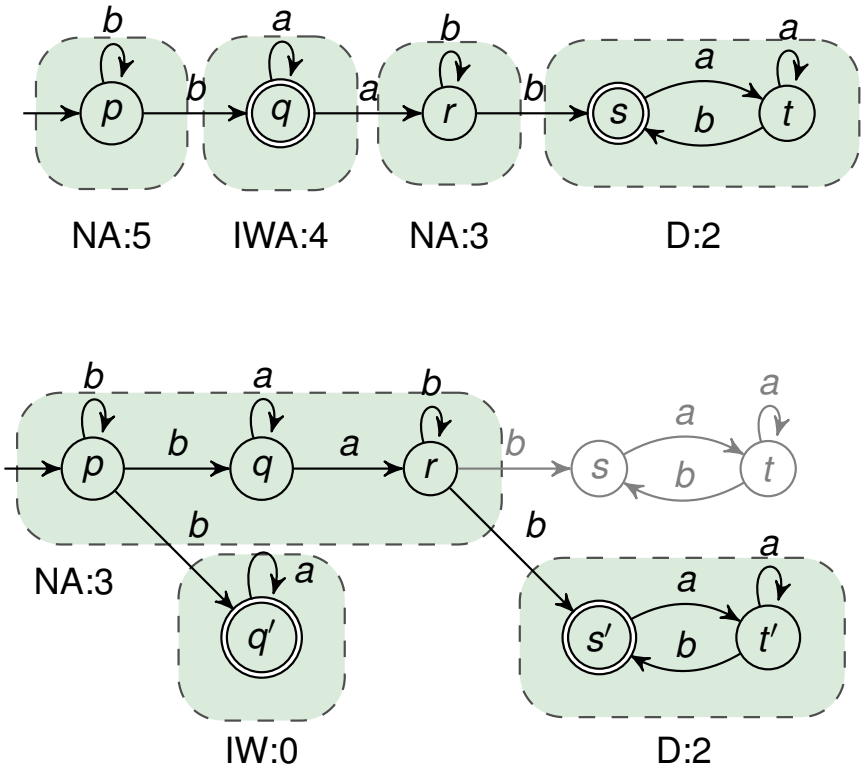




Complementation of Elevator Automata

Deelevation:

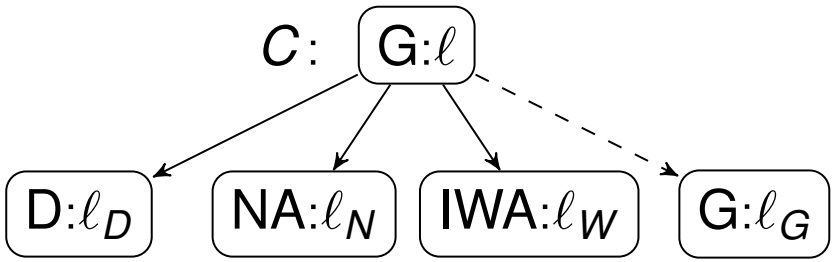
- Decreases the rank bound to 3
- At most $2|Q|$ states of the input automaton



Complementation of Non-Elevator Automata

- The technique can be generalized to non-elevator automata:
  - **G**: general SCC
- We can generalize the rules:

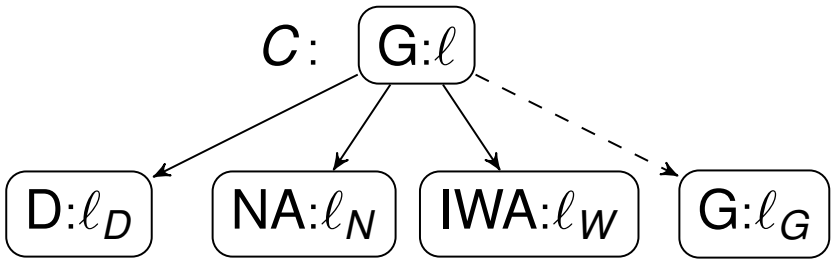
$$\ell = \max\{\ell_D, \ell_N + 1, \ell_W, \ell_G\} + 2|C \setminus Acc|$$



Complementation of Non-Elevator Automata

- The technique can be generalized to non-elevator automata:
  - **G**: general SCC
- We can generalize the rules:

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



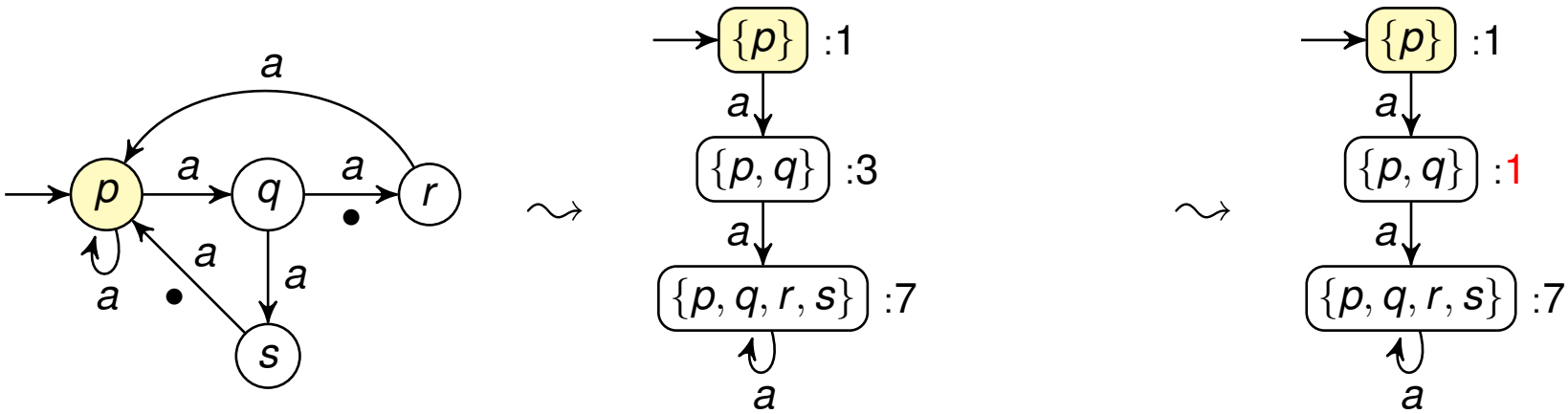
- Can we improve the rank bound for general SCCs?

Data Flow Analysis – Outer Macrostate Analysis

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

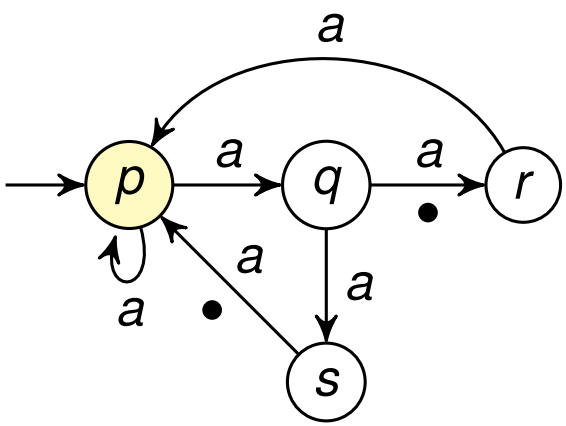
Data Flow Analysis – Outer Macrostate Analysis

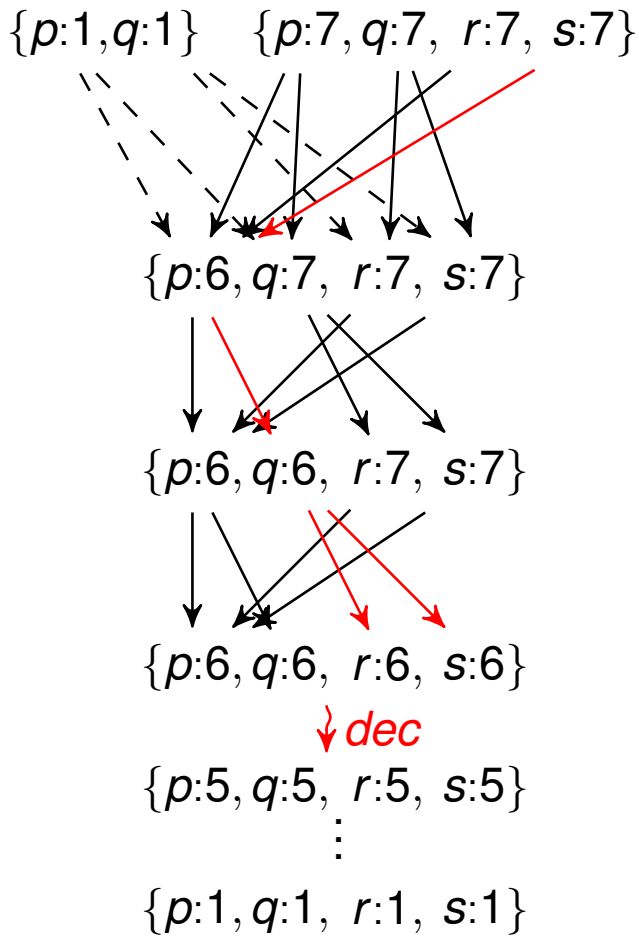
- Based on sizes of macrostates
- Bound for the smallest macrostate in every cycle
- Forward rank propagation



Data Flow Analysis — Inner Macrostate Analysis

- Based on ranks assigned to all states in a macrostate





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

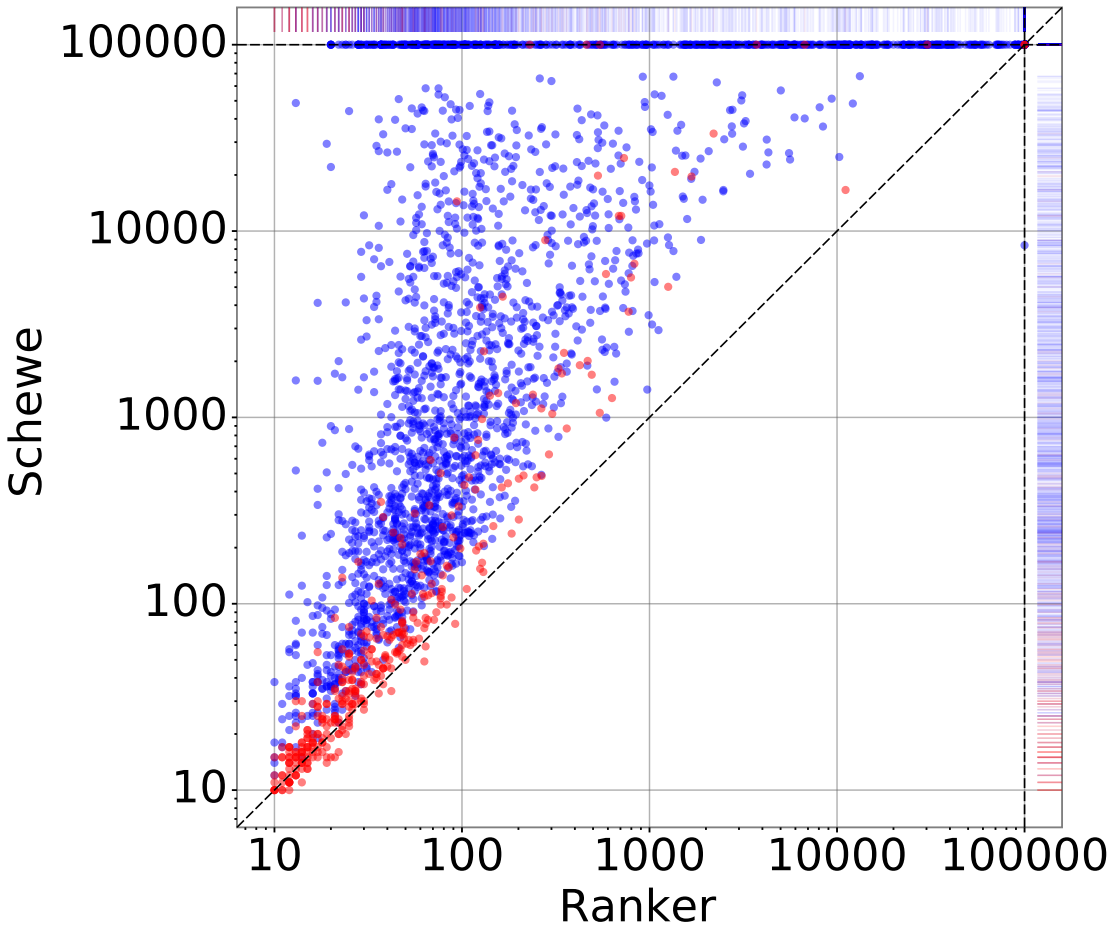
Experimental Evaluation

- 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

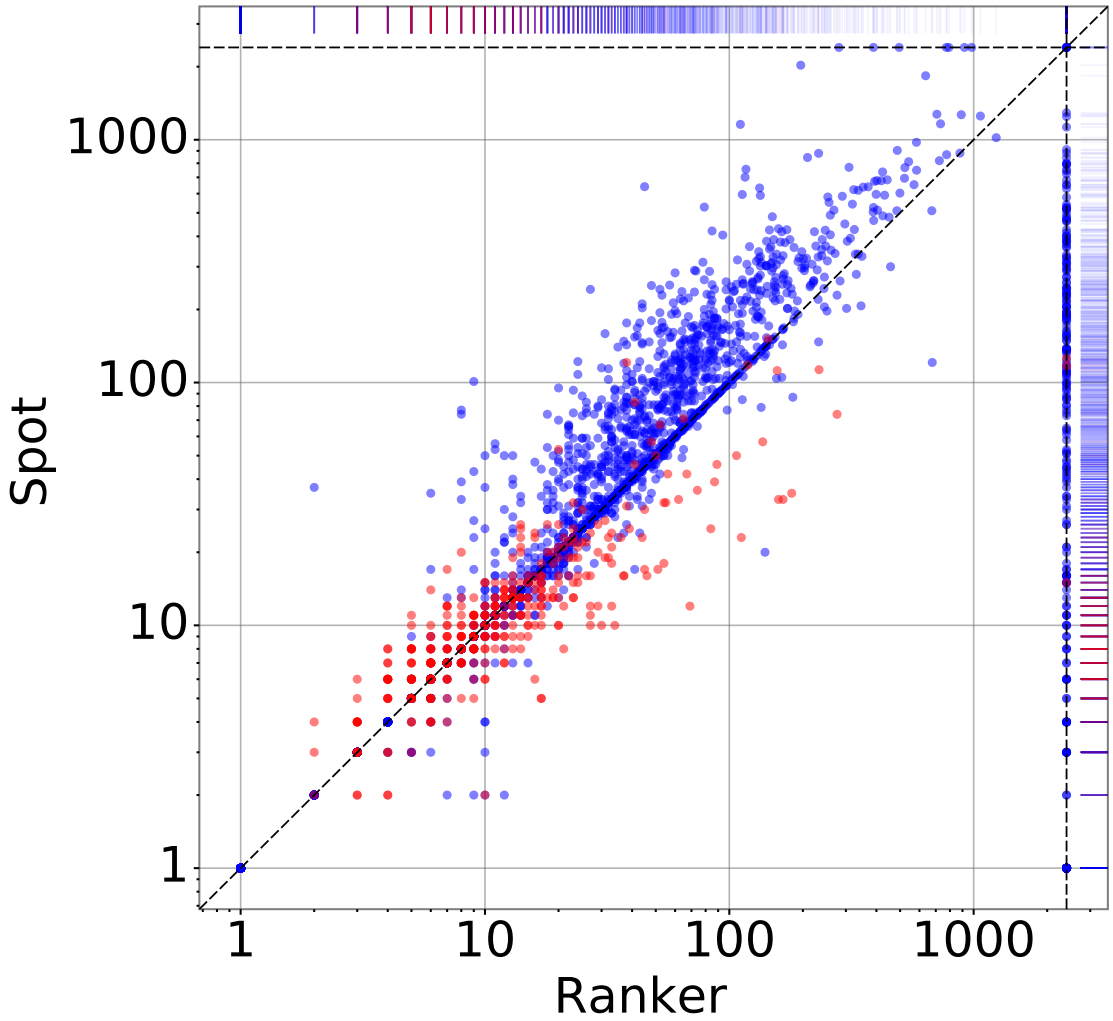
- Implemented in C++ within RANKER
- Compared with:
  - GOAL⚽ (SCHEWE, SAFRA, PITERMAN, FRIBOURG)
  - SPOT
  - LTL2DSTAR
  - SEMINATOR 2
  - ROLL

Experimental Evaluation – rank-based



- Number of states
- Logarithmic axes
- No postprocessing
- **blue**: random
- **red**: LTL

Experimental Evaluation – not rank-based



- Number of states
- Logarithmic axes
- With postprocessing
- **blue**: random
- **red**: LTL

Conclusion

- Definition of elevator automata
- Often occur in practice (90 % of LTL benchmark)
- Rank restriction based on the automaton structure
- Often exponential improvement
- Smaller complement in the majority of cases
- Competitive with other state-of-the-art tools

Conclusion

- Definition of elevator automata
- Often occur in practice (90 % of LTL benchmark)
- Rank restriction based on the automaton structure
- Often exponential improvement
- Smaller complement in the majority of cases
- Competitive with other state-of-the-art tools

Future work:

- Generalization to complementation of TELA
  - transition-based Emerson-Lei automata
- Language inclusion checking
- Decomposition-based complementation

Conclusion

- Definition of elevator automata
- Often occur in practice (90 % of LTL benchmark)
- Rank restriction based on the automaton structure
- Often exponential improvement
- Smaller complement in the majority of cases
- Competitive with other state-of-the-art tools

Future work:

- Generalization to complementation of TELA
  - transition-based Emerson-Lei automata
- Language inclusion checking
- Decomposition-based complementation

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)
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)

Experimental Evaluation – States

method	mean	median	wins	losses	timeouts
RANKER	3812 (4452 : 207)	79 (93 : 26)			279 (276 : 3)
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)
RANKER	47 (52 : 18)	22 (27 : 10)			279 (276 : 3)
PITERMAN ⚽	73 (82 : 22)	28 (34 : 14)	1435 (1124 : 311)	416 (360 : 56)	7 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)