Reducing (to) the Ranks:
Efficient Rank-based Büchi Automata Complementation

Vojtěch Havlena Ondřej Lengál

Brno University of Technology, Czech Republic

26 August 2021 (CONCUR'21)

Büchi Automata

- Automata over infinite words

Büchi Automata

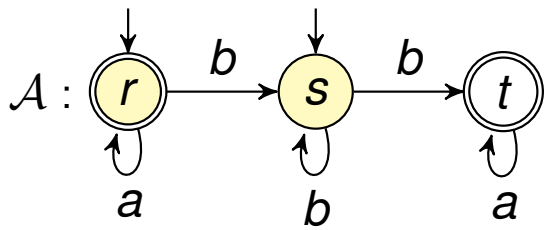
- Automata over infinite words
- $\mathcal{A} = (Q, \delta, I, F)$ over Σ
  - Q finite set of states
  - δ transition function; $\delta : Q \times \Sigma \to 2^Q$
  - $I \subseteq Q$ initial states
  - $F \subseteq Q$ accepting states

Büchi Automata

- Automata over infinite words
- $\mathcal{A} = (Q, \delta, I, F)$ over Σ
  - Q finite set of states
  - δ transition function; $\delta : Q \times \Sigma \to 2^Q$
  - $I \subseteq Q$ initial states
  - $F \subseteq Q$ accepting states
- Accepts via looping over accepting states
- Defines the class of ω-regular languages

Büchi Automata

- Automata over infinite words
- $\mathcal{A} = (Q, \delta, I, F)$ over Σ
  - Q finite set of states
  - δ transition function; $\delta : Q \times \Sigma \to 2^Q$
  - $I \subseteq Q$ initial states
  - $F \subseteq Q$ accepting states
- Accepts via looping over accepting states
- Defines the class of ω-regular languages



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

$abba^\omega \in \mathcal{L}(\mathcal{A})$

Motivation for Complementation

- Model checking of linear-time properties
  - property $\varphi \quad \rightsquigarrow \quad \mathcal{A}_\varphi$
  - system $S \quad \rightsquigarrow \quad \mathcal{A}_S$
  - checking $S \vDash \varphi \quad \rightsquigarrow \quad \mathcal{L}(\mathcal{A}_S) \subseteq \mathcal{L}(\mathcal{A}_\varphi) \quad \rightsquigarrow \quad \mathcal{L}(\mathcal{A}_S) \cap \mathcal{L}(\mathcal{A}_\varphi^{\complement}) = \emptyset$

Motivation for Complementation

- Model checking of linear-time properties
  - property $\varphi \quad \rightsquigarrow \quad \mathcal{A}_\varphi$
  - system $S \quad \rightsquigarrow \quad \mathcal{A}_S$
  - checking $S \models \varphi \quad \rightsquigarrow \quad \mathcal{L}(\mathcal{A}_S) \subseteq \mathcal{L}(\mathcal{A}_\varphi) \quad \rightsquigarrow \quad \mathcal{L}(\mathcal{A}_S) \cap \mathcal{L}(\mathcal{A}_\varphi^{\complement}) = \emptyset$
- Termination analysis: Ultimate Automizer [HeizmannHP]
  - removing traces with proved termination
  - difference automaton

Motivation for Complementation

- Model checking of linear-time properties
  - property $\varphi \quad \rightsquigarrow \quad \mathcal{A}_\varphi$
  - system $S \quad \rightsquigarrow \quad \mathcal{A}_S$
  - checking $S \models \varphi \quad \rightsquigarrow \quad \mathcal{L}(\mathcal{A}_S) \subseteq \mathcal{L}(\mathcal{A}_\varphi) \quad \rightsquigarrow \quad \mathcal{L}(\mathcal{A}_S) \cap \mathcal{L}(\mathcal{A}_\varphi^{\complement}) = \emptyset$
- Termination analysis: Ultimate Automizer [HeizmannHP]
  - removing traces with proved termination
  - difference automaton
- Decision procedures:
  - S1S: MSO over $(\omega, 0, +1)$
  - ETL: extended temporal logic
  - QPTL: quantified propositional temporal logic
  - FO over Sturmian words

Büchi Automata Complementation

- More involved than for NFAs (cannot be determinized)
- Lower bound $(0.76n)^n$ [Yan'06]

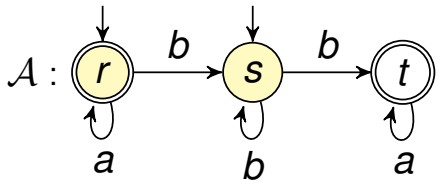
Büchi Automata Complementation

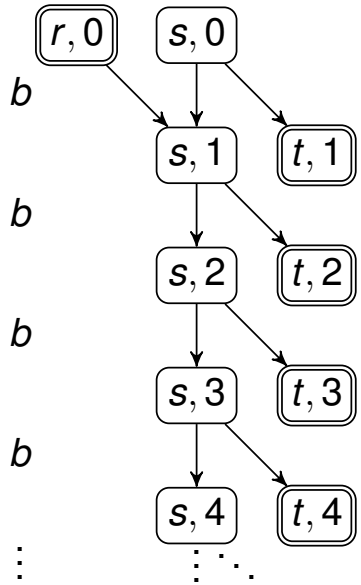
- More involved than for NFAs (cannot be determinized)
- Lower bound $(0.76n)^n$ [Yan'06]

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

Rank-based Complementation

- Run DAG $\mathcal{G}_\alpha$ of a word α
  - all runs on α in $\mathcal{A}$
  - vertices: (s, ℓ); $s \in Q, \ell \in \mathbb{N}$

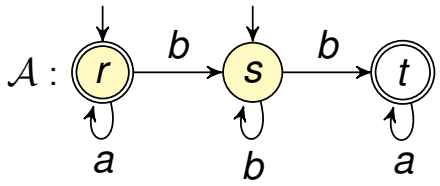




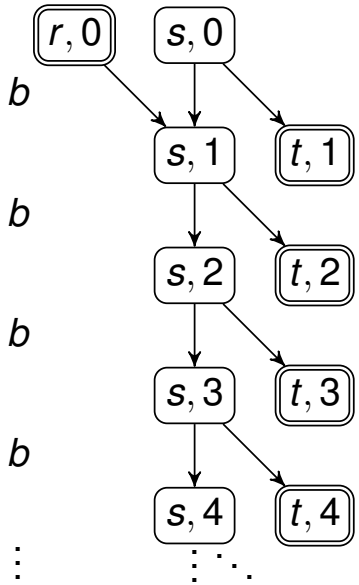
$b^\omega \notin \mathcal{L}(\mathcal{A})$

Rank-based Complementation

- Run DAG $\mathcal{G}_\alpha$ of a word α
  - all runs on α in $\mathcal{A}$
  - vertices: (s, ℓ); $s \in Q, \ell \in \mathbb{N}$



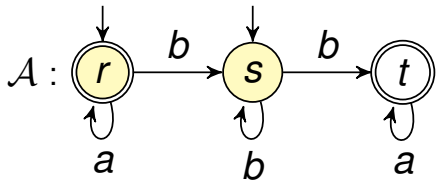
- Ranking procedure (set $i := 0$)
  1. assign rank i to vertices with finitely many successors and remove them from $\mathcal{G}_\alpha$
  2. assign rank $i + 1$ to vertices that cannot reach F and remove them from $\mathcal{G}_\alpha$
  3. $i := i + 2$; repeat while $i \leq 2|Q|$



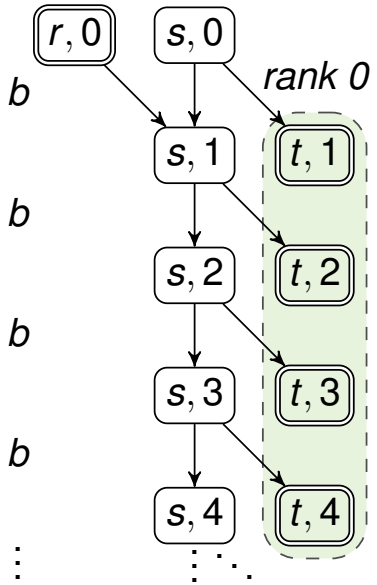
$b^\omega \notin \mathcal{L}(\mathcal{A})$

Rank-based Complementation

- Run DAG $\mathcal{G}_\alpha$ of a word α
  - all runs on α in $\mathcal{A}$
  - vertices: (s, ℓ); $s \in Q, \ell \in \mathbb{N}$



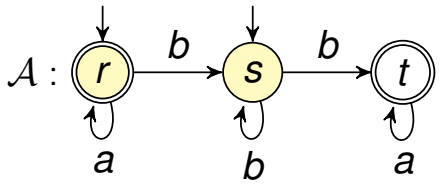
- Ranking procedure (set $i := 0$)
  1. assign rank i to vertices with finitely many successors and remove them from $\mathcal{G}_\alpha$
  2. assign rank $i + 1$ to vertices that cannot reach F and remove them from $\mathcal{G}_\alpha$
  3. $i := i + 2$; repeat while $i \leq 2|Q|$



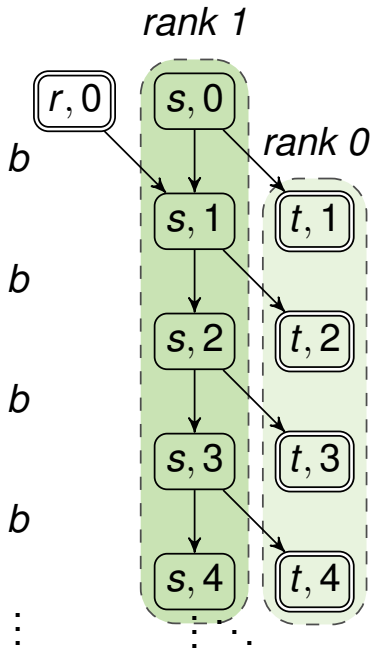
$b^\omega \notin \mathcal{L}(\mathcal{A})$

Rank-based Complementation

- Run DAG $\mathcal{G}_\alpha$ of a word α
  - all runs on α in $\mathcal{A}$
  - vertices: (s, ℓ); $s \in Q, \ell \in \mathbb{N}$



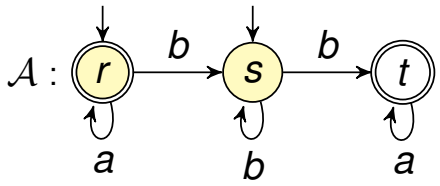
- Ranking procedure (set $i := 0$)
  1. assign rank i to vertices with finitely many successors and remove them from $\mathcal{G}_\alpha$
  2. assign rank $i + 1$ to vertices that cannot reach F and remove them from $\mathcal{G}_\alpha$
  3. $i := i + 2$; repeat while $i \leq 2|Q|$



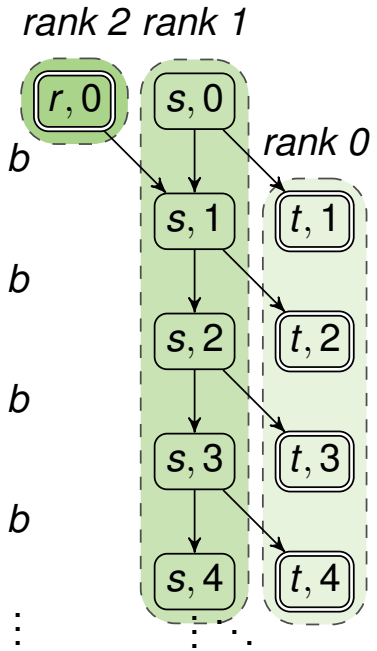
$b^\omega \notin \mathcal{L}(\mathcal{A})$

Rank-based Complementation

- Run DAG $\mathcal{G}_\alpha$ of a word α
  - all runs on α in $\mathcal{A}$
  - vertices: (s, ℓ); $s \in Q, \ell \in \mathbb{N}$



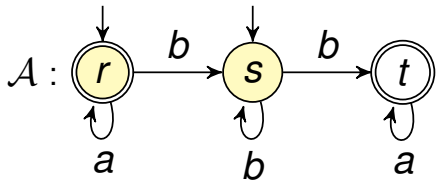
- Ranking procedure (set $i := 0$)
  1. assign rank i to vertices with finitely many successors and remove them from $\mathcal{G}_\alpha$
  2. assign rank $i + 1$ to vertices that cannot reach F and remove them from $\mathcal{G}_\alpha$
  3. $i := i + 2$; repeat while $i \leq 2|Q|$



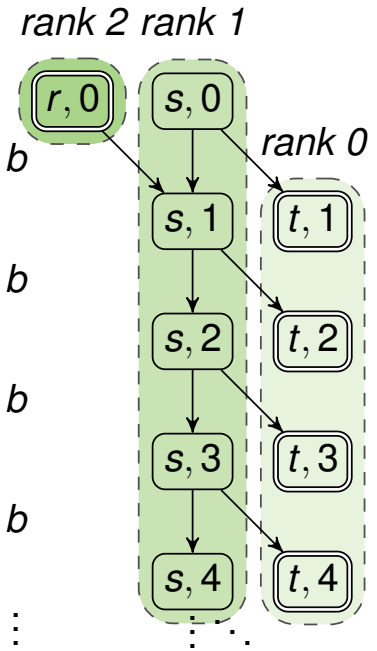
$b^\omega \notin \mathcal{L}(\mathcal{A})$

Rank-based Complementation

- Run DAG $\mathcal{G}_\alpha$ of a word α
  - all runs on α in $\mathcal{A}$
  - vertices: (s, ℓ); $s \in Q, \ell \in \mathbb{N}$



- Ranking procedure (set $i := 0$)
  1. assign rank i to vertices with finitely many successors and remove them from $\mathcal{G}_\alpha$
  2. assign rank $i + 1$ to vertices that cannot reach F and remove them from $\mathcal{G}_\alpha$
  3. $i := i + 2$; repeat while $i \leq 2|Q|$



Lemma [Kupferman,Vardi'01]

If $\alpha \notin \mathcal{L}(\mathcal{A})$ then $\forall v \in \mathcal{G}_\alpha : \mathit{rank}(v) \leq 2|Q|$.

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

Rank-based Complementation *Cont.*

- Nondeterministically guesses run DAG ranks [Schewe'09]

Rank-based Complementation *Cont.*

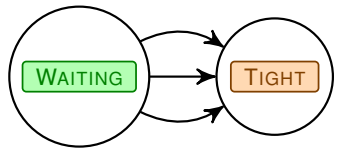
- Nondeterministically guesses run DAG ranks [Schewe'09]
- Macrostates (S, O, f, i); accepting macrostates $(\cdot, \emptyset, \cdot, \cdot)$ (omit i)
  - S tracks all runs of $\mathcal{A}$ (determinization of NFAs)
  - O tracks all runs with an even rank (since a breakpoint with $O = \emptyset$)
  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

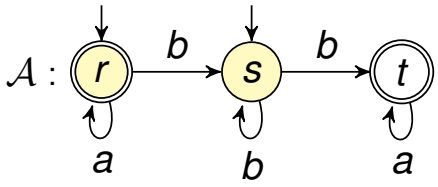
- Nondeterministically guesses run DAG ranks [Schewe'09]
- Macrostates (S, O, f, i); accepting macrostates $(\cdot, \emptyset, \cdot, \cdot)$ (omit i)
  - S tracks all runs of $\mathcal{A}$ (determinization of NFAs)
  - O tracks all runs with an even rank (since a breakpoint with $O = \emptyset$)
  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

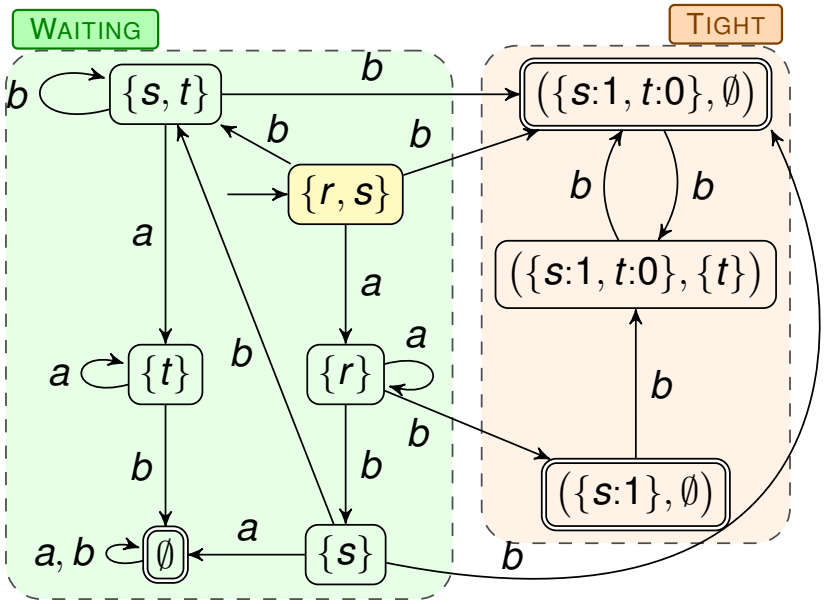
Rank-based Complementation *Cont.*

- 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)
- WAITING and TIGHT part
  - in WAITING guess the point from which all successor rankings are tight (only S-part)
  - in TIGHT track tight rankings

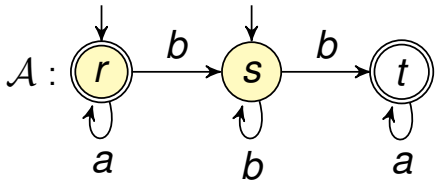


Rank-based Complementation *Example*

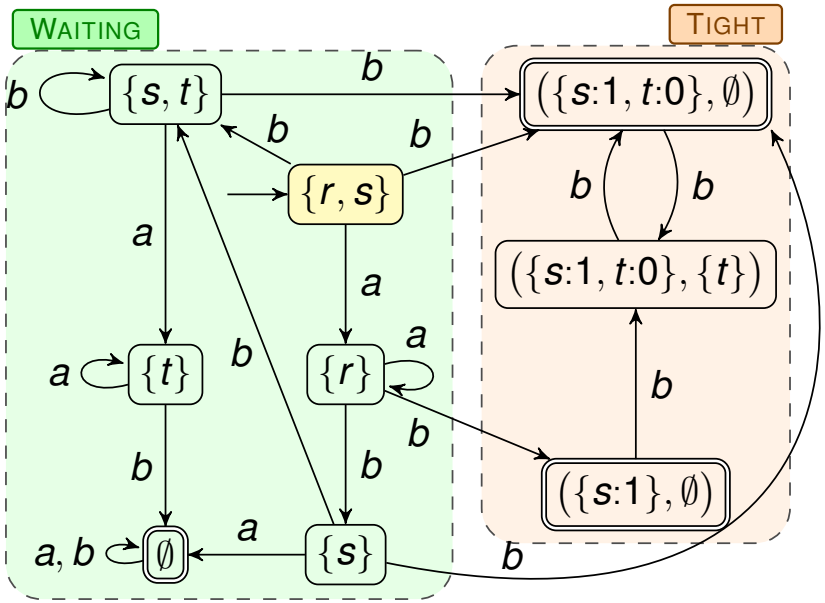




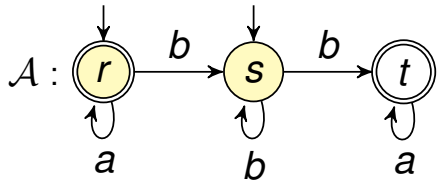
Rank-based Complementation *Example*

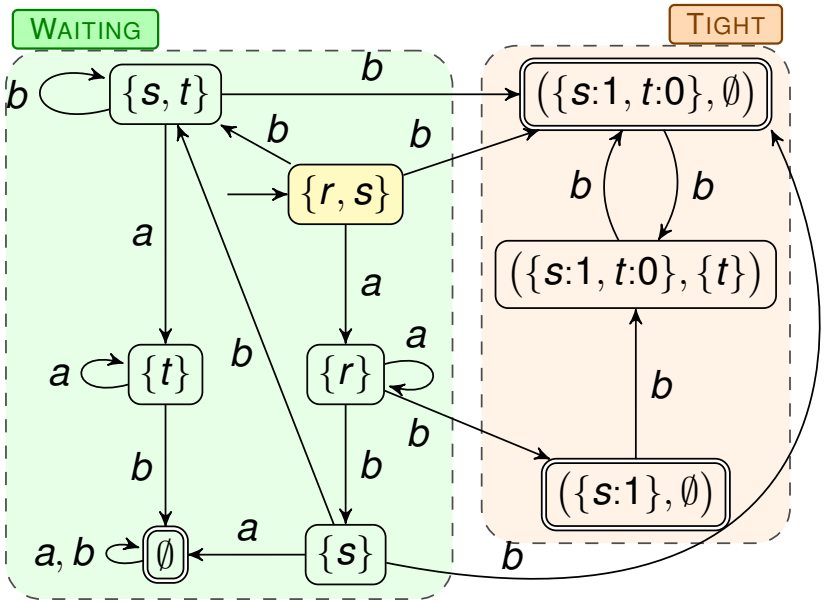


- $(\{s{:}1, t{:}0\}, \emptyset) \xrightarrow{b} (S', O', f')$
  - $S' = \delta(\{s,t\}, b) = \{s,t\}$
  - $f'(s) \leq f(s)$, $f'(t) \leq f(s)$,
    $f'(t)$ is even $\implies \{s{:}1, t{:}0\}$
  - $O' = \{t\}$ $\quad (O' = S' \cap \mathit{even}(f'))$
  - $(\{s{:}1, t{:}0\}, \{t\})$



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

Rank-based Complementation *Problems*

- Highly nondeterministic
  - when to switch from WAITING to TIGHT?
  - how to decrease ranks in TIGHT?
  - $\rightsquigarrow$ many different (redundant) successors!

Rank-based Complementation *Problems*

- Highly nondeterministic
  - when to switch from WAITING to TIGHT?
  - how to decrease ranks in TIGHT?
  - $\rightsquigarrow$ many different (redundant) successors!
- Tight rank bound $b = 2|Q| - 1$ is often too coarse
  - unnecessarily high bound $\rightsquigarrow$ many redundant states are generated
  - state space explosion $\rightsquigarrow$ combinatorial explosion wrt. b

Rank-based Complementation *Problems*

- Highly nondeterministic
  - when to switch from WAITING to TIGHT?
  - how to decrease ranks in TIGHT?
  - $\rightsquigarrow$ many different (redundant) successors!
- Tight rank bound $b = 2|Q| - 1$ is often too coarse
  - unnecessarily high bound $\rightsquigarrow$ many redundant states are generated
  - state space explosion $\rightsquigarrow$ combinatorial explosion wrt. b

Reduce nondeterminism and keep the ranks as small as possible!

Rank-based Complementation *Problems*

- Highly nondeterministic
  - when to switch from WAITING to TIGHT?
  - how to decrease ranks in TIGHT?
  - $\rightsquigarrow$ many different (redundant) successors!
- Tight rank bound $b = 2|Q| - 1$ is often too coarse
  - unnecessarily high bound $\rightsquigarrow$ many redundant states are generated
  - state space explosion $\rightsquigarrow$ combinatorial explosion wrt. b

Reduce nondeterminism and keep the ranks as small as possible!

- Lightweight optimizations
- Based on the notion of super-tight runs
  - a run on $\alpha \in \mathcal{L}(\mathcal{A}^{\complement})$ using as small ranks as possible
  - enough to preserve only super-tight runs

DELAY Optimization

- More super-tight runs that differ on the position of the transition from WAITING into TIGHT:
  - $\boxed{S_0} \xrightarrow{a} \cdots \xrightarrow{a} \boxed{S_k} \xrightarrow{a} \boxed{(S_{k+1}, \cdot, f_{k+1})} \xrightarrow{a} \cdots$ a^ω
  - $\boxed{S_0} \xrightarrow{a} \cdots \xrightarrow{a} \boxed{S_k} \xrightarrow{a} \cdots \xrightarrow{a} \boxed{S_\ell} \xrightarrow{a} \boxed{(S_{\ell+1}, \cdot, f_{\ell+1})} \xrightarrow{a} \cdots$

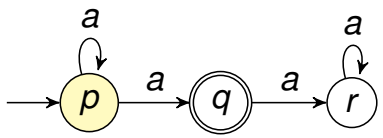
DELAY Optimization

- More super-tight runs that differ on the position of the transition from WAITING into TIGHT:
  - $\boxed{S_0} \xrightarrow{a} \cdots \xrightarrow{a} \boxed{S_k} \xrightarrow{a} \boxed{(S_{k+1}, \cdot, f_{k+1})} \xrightarrow{a} \cdots$ a^ω
  - $\boxed{S_0} \xrightarrow{a} \cdots \xrightarrow{a} \boxed{S_k} \xrightarrow{a} \cdots \xrightarrow{a} \boxed{S_\ell} \xrightarrow{a} \boxed{(S_{\ell+1}, \cdot, f_{\ell+1})} \xrightarrow{a} \cdots$
- Postpone transitions to the tight part
  - generate transitions only when a cycle is closed in WAITING
    - similar to the **C3** (cycle) condition in partial-order reduction (POR).
  - reduces the number of transitions to TIGHT

DELAY Optimization

- More super-tight runs that differ on the position of the transition from WAITING into TIGHT:
  - $\boxed{S_0} \xrightarrow{a} \cdots \xrightarrow{a} \boxed{S_k} \xrightarrow{a} \boxed{(S_{k+1}, \cdot, f_{k+1})} \xrightarrow{a} \cdots$ a^ω
  - $\boxed{S_0} \xrightarrow{a} \cdots \xrightarrow{a} \boxed{S_k} \xrightarrow{a} \cdots \xrightarrow{a} \boxed{S_\ell} \xrightarrow{a} \boxed{(S_{\ell+1}, \cdot, f_{\ell+1})} \xrightarrow{a} \cdots$
- Postpone transitions to the tight part
  - generate transitions only when a cycle is closed in WAITING
    - similar to the **C3** (cycle) condition in partial-order reduction (POR).
  - reduces the number of transitions to TIGHT

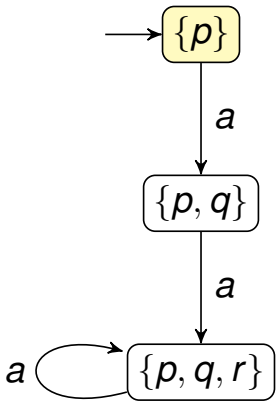
Example



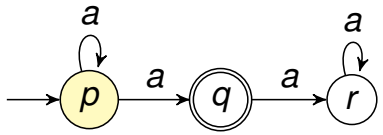
DELAY Optimization

- More super-tight runs that differ on the position of the transition from WAITING into TIGHT:
  - $S_0 \xrightarrow{a} \cdots \xrightarrow{a} S_k \xrightarrow{a} (S_{k+1}, \cdot, f_{k+1}) \xrightarrow{a} \cdots$ a^ω
  - $S_0 \xrightarrow{a} \cdots \xrightarrow{a} S_k \xrightarrow{a} \cdots \xrightarrow{a} S_\ell \xrightarrow{a} (S_{\ell+1}, \cdot, f_{\ell+1}) \xrightarrow{a} \cdots$
- Postpone transitions to the tight part
  - generate transitions only when a cycle is closed in WAITING
    - similar to the **C3** (cycle) condition in partial-order reduction (POR).
  - reduces the number of transitions to TIGHT

Example



$\rightsquigarrow \{p\}, \{p, q\}$

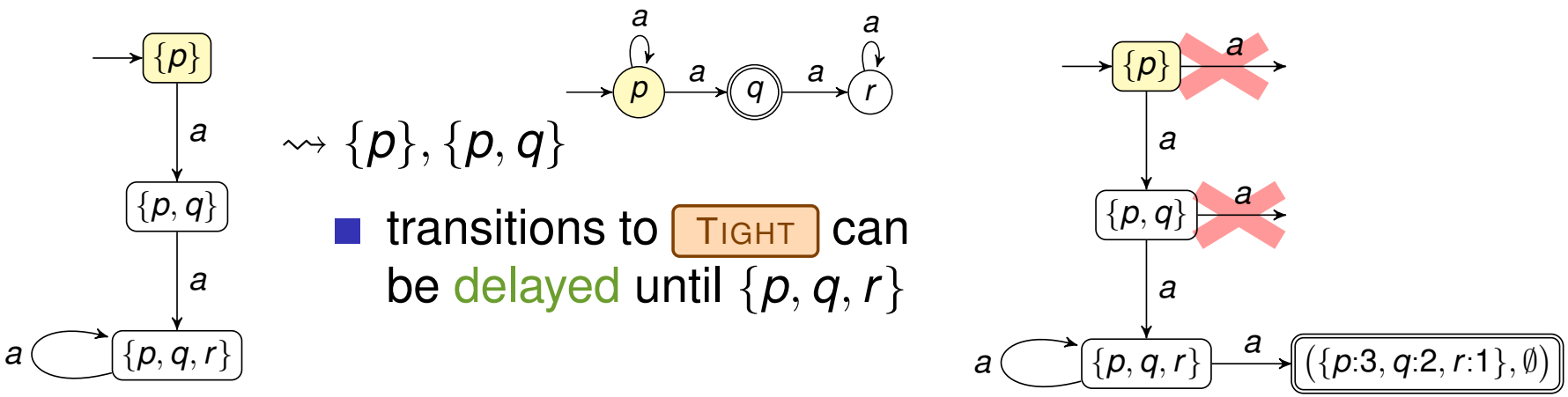


- transitions to TIGHT can be delayed until $\{p, q, r\}$

DELAY Optimization

- More super-tight runs that differ on the position of the transition from WAITING into TIGHT:
  - $S_0 \xrightarrow{a} \cdots \xrightarrow{a} S_k \xrightarrow{a} (S_{k+1}, \cdot, f_{k+1}) \xrightarrow{a} \cdots$ a^ω
  - $S_0 \xrightarrow{a} \cdots \xrightarrow{a} S_k \xrightarrow{a} \cdots \xrightarrow{a} S_\ell \xrightarrow{a} (S_{\ell+1}, \cdot, f_{\ell+1}) \xrightarrow{a} \cdots$
- Postpone transitions to the tight part
  - generate transitions only when a cycle is closed in WAITING
    - similar to the **C3** (cycle) condition in partial-order reduction (POR).
  - reduces the number of transitions to TIGHT

Example



- transitions to TIGHT can be delayed until $\{p, q, r\}$

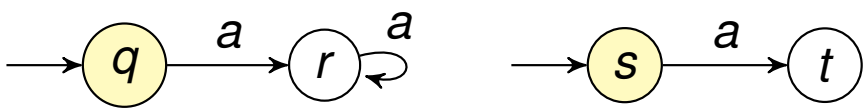
SuccRank Optimization I

- Reasoning about Waiting
  - Waiting defines the skeleton (S-part) of Tight
  - sizes of reachable macrostates restrict the maximum rank
  - bounded by the size of a max ∞-often reachable macrostate $\lceil \cdot \rceil$
    - $(S, O, f) \rightsquigarrow \mathit{rank}(f) \leq 2\lceil S \rceil - 1$

SuccRank Optimization I

- Reasoning about Waiting
  - Waiting defines the skeleton (S-part) of Tight
  - sizes of reachable macrostates restrict the maximum rank
  - bounded by the size of a max ∞-often reachable macrostate $\lceil \cdot \rceil$
    - $(S, O, f) \rightsquigarrow \mathit{rank}(f) \leq 2\lceil S \rceil - 1$

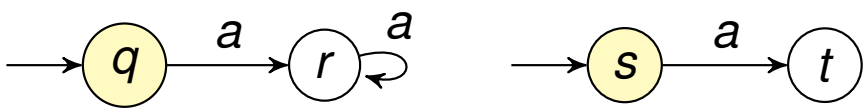
Example

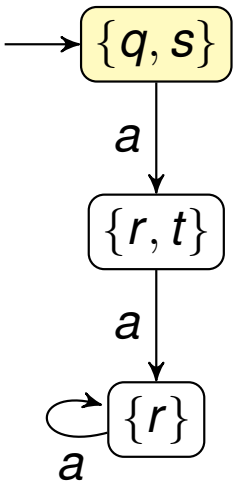


SuccRank Optimization I

- Reasoning about Waiting
  - Waiting defines the skeleton (S-part) of Tight
  - sizes of reachable macrostates restrict the maximum rank
  - bounded by the size of a max ∞-often reachable macrostate $\lceil\cdot\rceil$
    - $(S, O, f) \rightsquigarrow \mathit{rank}(f) \leq 2\lceil S \rceil - 1$

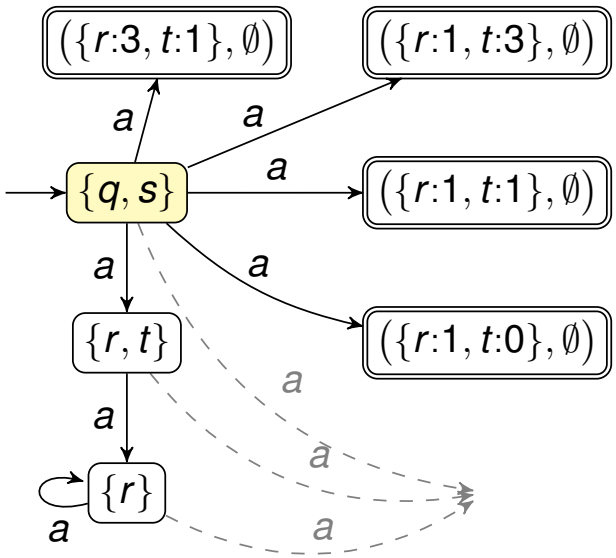
Example





$\rightsquigarrow \{r, t\}$

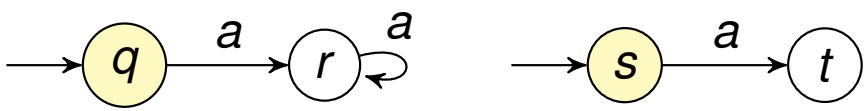
- maximal ∞-reach macrostate: $\{r\}$
- remove $(\{r, t\}, \cdot, f)$ s.t. $\mathit{rank}(f) > 2|\{r\}| - 1 = 1$

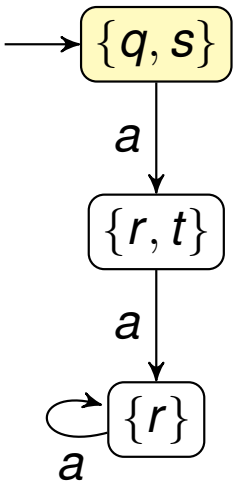


SuccRank Optimization I

- Reasoning about Waiting
  - Waiting defines the skeleton (S-part) of Tight
  - sizes of reachable macrostates restrict the maximum rank
  - bounded by the size of a max ∞-often reachable macrostate $\lceil\cdot\rceil$
    - $(S, O, f) \rightsquigarrow \mathit{rank}(f) \leq 2\lceil S \rceil - 1$

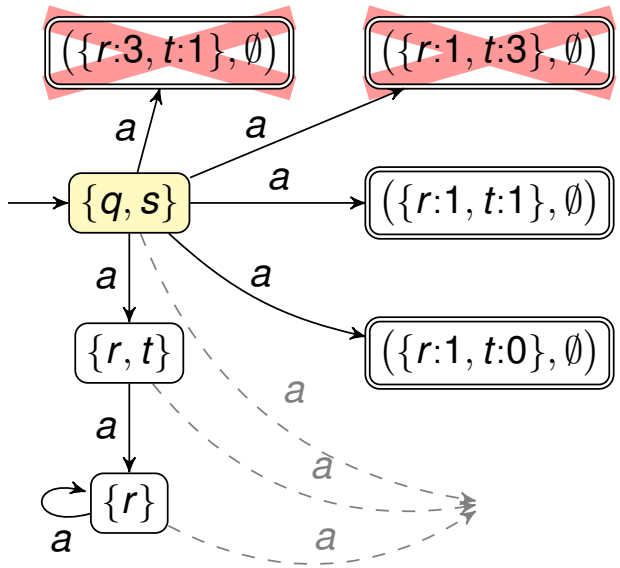
Example





$\rightsquigarrow \{r, t\}$

- maximal ∞-reach macrostate: $\{r\}$
- remove $(\{r, t\}, \cdot, f)$ s.t. $\mathit{rank}(f) > 2|\{r\}| - 1 = 1$



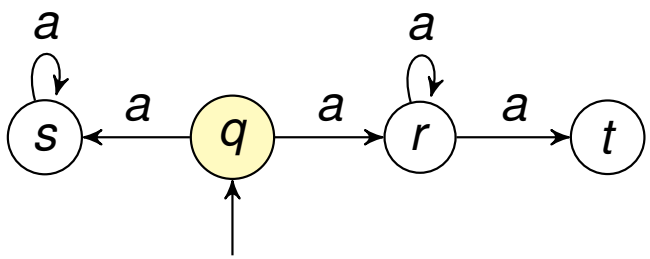
SuccRank Optimization II

- Refine with a minimal ∞-often reachable macrostate $\lfloor \cdot \rfloor$
  - (S, O, f)
  - $q \in S \rightsquigarrow$ at most $\lceil S \rceil - \lfloor q \rfloor$ states with higher rank than $f(q)$
  - reduces the value of $rank(f)$; $rank(f) \leq f(q) + 2(\lceil S \rceil - \lfloor q \rfloor)$

SuccRank Optimization II

- Refine with a minimal ∞-often reachable macrostate $\lfloor \cdot \rfloor$
  - (S, O, f)
  - $q \in S \rightsquigarrow$ at most $\lceil S \rceil - \lfloor q \rfloor$ states with higher rank than $f(q)$
  - reduces the value of $rank(f)$; $rank(f) \leq f(q) + 2(\lceil S \rceil - \lfloor q \rfloor)$

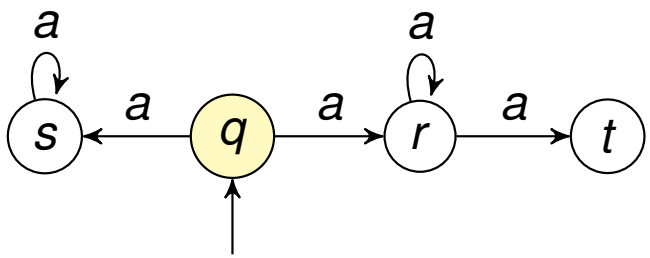
Example

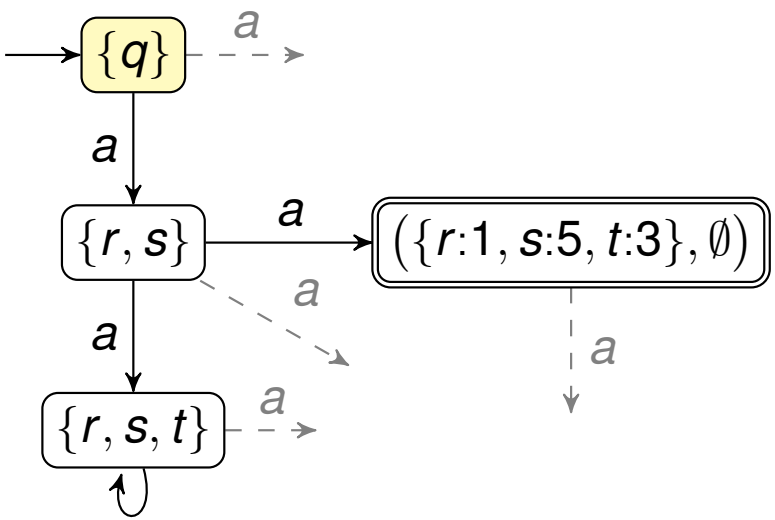


SuccRank Optimization II

- Refine with a minimal ∞-often reachable macrostate $\lfloor \cdot \rfloor$
  - (S, O, f)
  - $q \in S \rightsquigarrow$ at most $\lceil S \rceil - \lfloor q \rfloor$ states with higher rank than $f(q)$
  - reduces the value of $rank(f)$; $rank(f) \leq f(q) + 2(\lceil S \rceil - \lfloor q \rfloor)$

Example





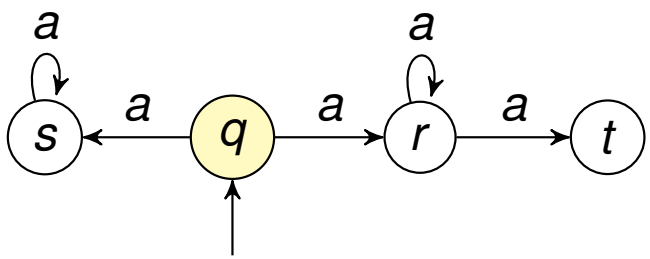
$\rightsquigarrow (\{r{:}1, s{:}5, t{:}3\}, \emptyset)$

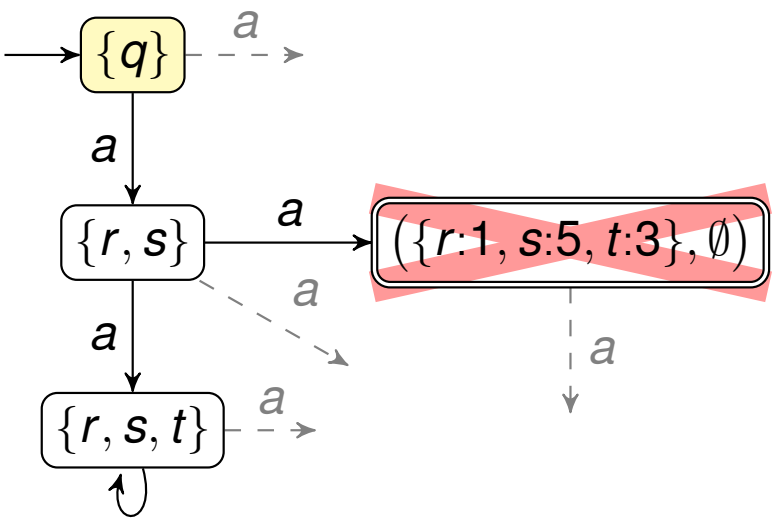
- $\{r\} \to^* \{r, t\}$
- $\lceil \{r, s, t\} \rceil = 3$, $\lfloor r \rfloor = 2$
- $rank(f) \leq 1 + 2 = 3$
- redundant macrostate

SuccRank Optimization II

- Refine with a minimal ∞-often reachable macrostate $\lfloor \cdot \rfloor$
  - (S, O, f)
  - $q \in S \rightsquigarrow$ at most $\lceil S \rceil - \lfloor q \rfloor$ states with higher rank than $f(q)$
  - reduces the value of $rank(f)$; $rank(f) \leq f(q) + 2(\lceil S \rceil - \lfloor q \rfloor)$

Example





$\rightsquigarrow (\{r{:}1, s{:}5, t{:}3\}, \emptyset)$

- $\{r\} \to^* \{r, t\}$
- $\lceil \{r, s, t\} \rceil = 3$, $\lfloor r \rfloor = 2$
- $rank(f) \leq 1 + 2 = 3$
- redundant macrostate

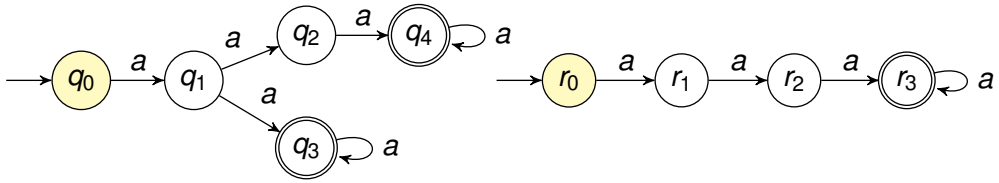
RankSim Optimization

- Relation between state-ranks inside each macrostate
- For each macrostate (S, O, f) in every super-tight run:
  - $p \preceq_{ors} r \ \wedge f(p)$ odd $\wedge f(r)$ odd $\implies f(p) \leq f(r)$
    - ors: odd rank simulation

RankSim Optimization

- Relation between state-ranks inside each macrostate
- For each macrostate (S, O, f) in every super-tight run:
  - $p \preceq_{ors} r \ \wedge f(p)$ odd $\wedge f(r)$ odd $\implies f(p) \leq f(r)$
    - ors: odd rank simulation

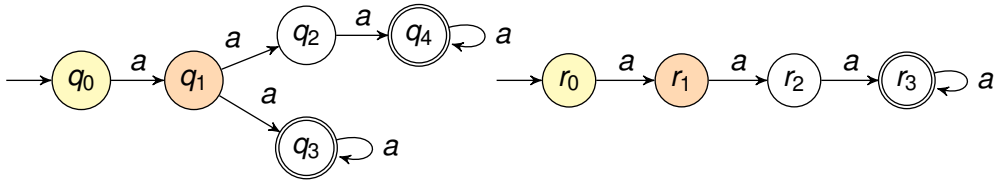
Example

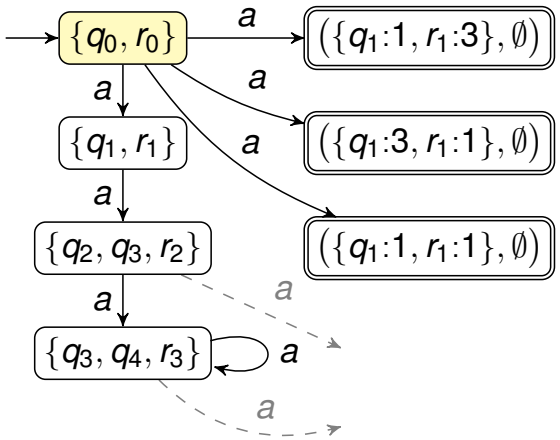


RankSim Optimization

- Relation between state-ranks inside each macrostate
- For each macrostate (S, O, f) in every super-tight run:
  - $p \preceq_{ors} r \ \wedge f(p)$ odd $\wedge f(r)$ odd $\implies f(p) \leq f(r)$
    - ors: odd rank simulation

Example





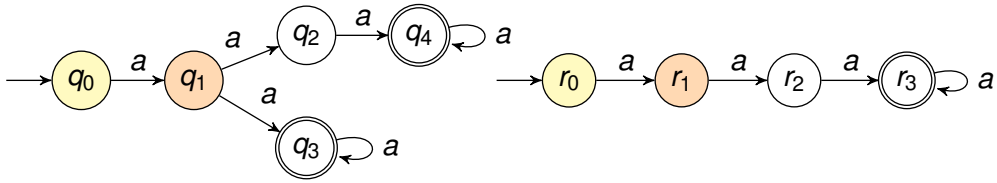
$\rightsquigarrow \big(\{q_1{:}1, r_1{:}3\}, \emptyset\big), \big(\{q_1{:}3, r_1{:}1\}, \emptyset\big)$

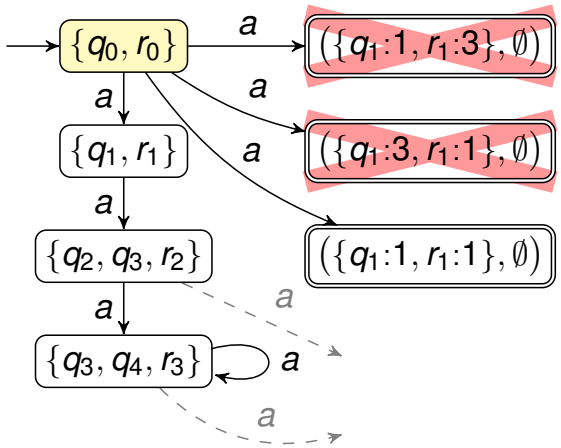
- $q_1 \preceq_{ors} r_1$ and $r_1 \preceq_{ors} q_1$
- $f(q_1) = f(r_1)$

RankSim Optimization

- Relation between state-ranks inside each macrostate
- For each macrostate (S, O, f) in every super-tight run:
  - $p \preceq_{ors} r \ \wedge f(p)$ odd $\wedge f(r)$ odd $\implies f(p) \leq f(r)$
    - ors: odd rank simulation

Example





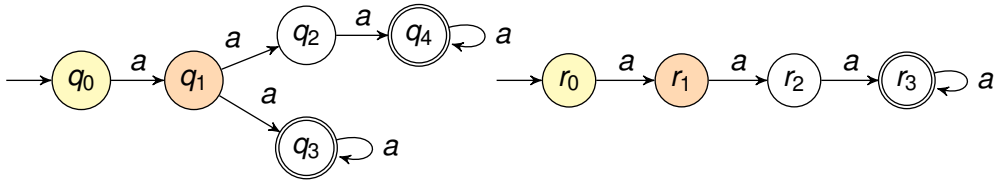
$\rightsquigarrow \big(\{q_1{:}1, r_1{:}3\}, \emptyset\big), \big(\{q_1{:}3, r_1{:}1\}, \emptyset\big)$

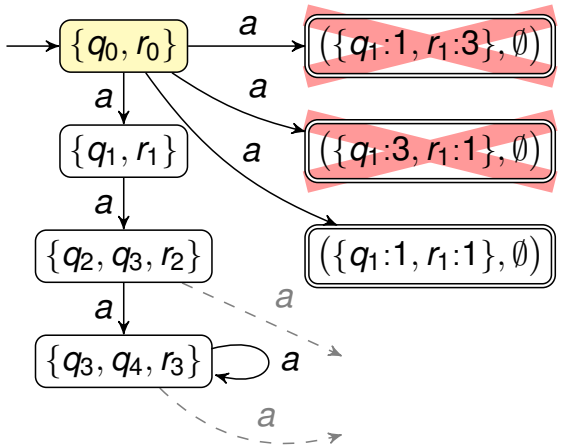
- $q_1 \preceq_{ors} r_1$ and $r_1 \preceq_{ors} q_1$
- $f(q_1) = f(r_1)$

RankSim Optimization

- Relation between state-ranks inside each macrostate
- For each macrostate (S, O, f) in every super-tight run:
  - $p \preceq_{ors} r \ \wedge f(p)$ odd $\wedge f(r)$ odd $\implies f(p) \leq f(r)$
    - ors: odd rank simulation

Example





$\rightsquigarrow \big(\{q_1{:}1, r_1{:}3\}, \emptyset\big), \big(\{q_1{:}3, r_1{:}1\}, \emptyset\big)$

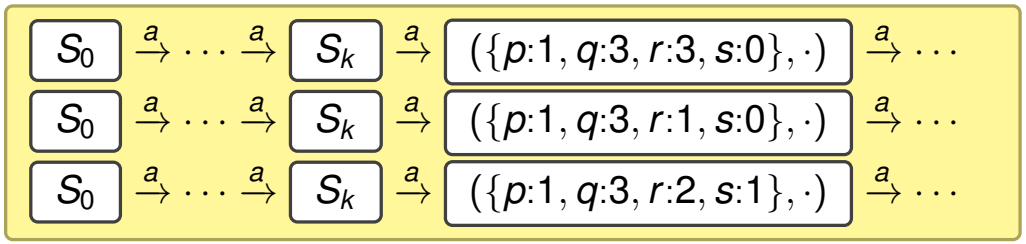
- $q_1 \preceq_{ors} r_1$ and $r_1 \preceq_{ors} q_1$
- $f(q_1) = f(r_1)$

- underapproximation $\preceq_R$ of $\preceq_{ors} \rightsquigarrow$ lfp computation from $\preceq_{di}$
  - step: $\forall a \in \Sigma : (\delta(p, a) \setminus F) \preceq_R (\delta(q, a) \setminus F)$ then $p \preceq_R q$

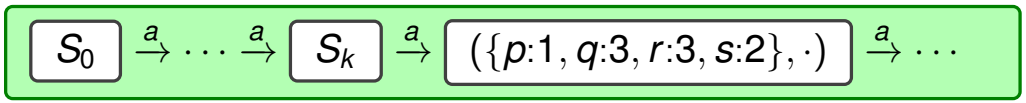
MaxRank Optimization

- (inspired by RedAvgOut from [Schewe'09])
- Represents several runs (incl. super-tight) using a maximal run

Runs



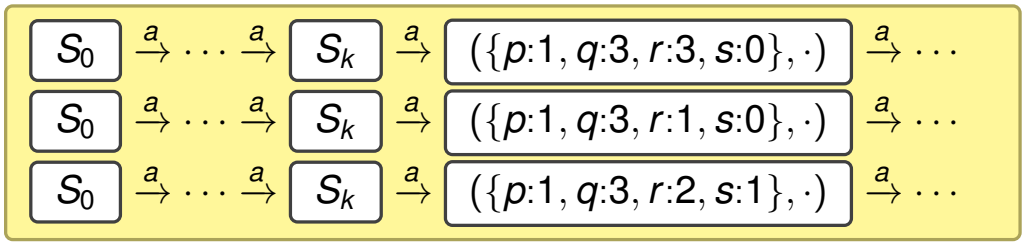
Maximal run



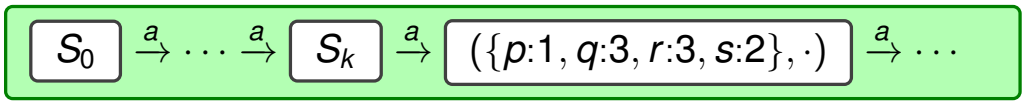
MaxRank Optimization

- (inspired by RedAvgOut from [Schewe'09])
- Represents several runs (incl. super-tight) using a maximal run

Runs



Maximal run

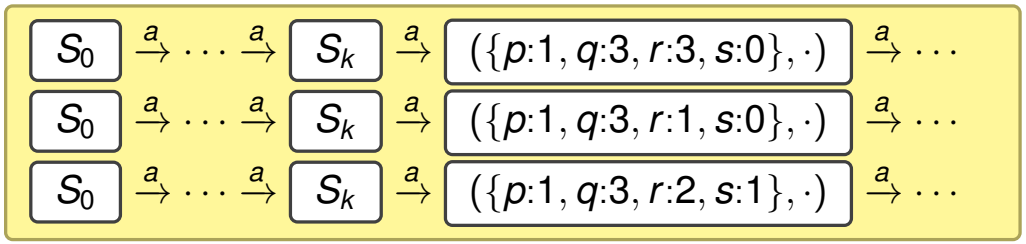


- Reduces the number of nondeterministic choices
  - decrease ranks of all possible states in the O-set, or
  - continue to the maximal macrostate
- Removing macrostates using the previous optimizations

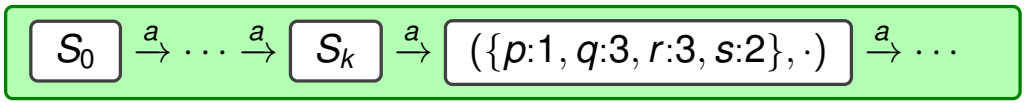
MaxRank Optimization

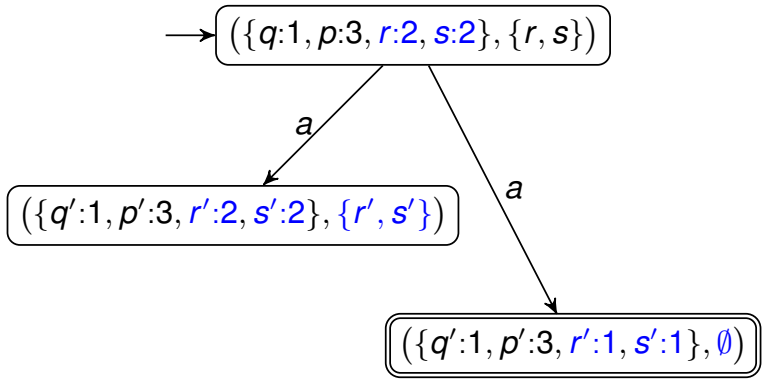
- (inspired by RedAvgOut from [Schewe'09])
- Represents several runs (incl. super-tight) using a maximal run

Runs



Maximal run





- Reduces the number of nondeterministic choices
  - decrease ranks of all possible states in the O-set, or
  - continue to the maximal macrostate
- Removing macrostates using the previous optimizations

Experimental Evaluation

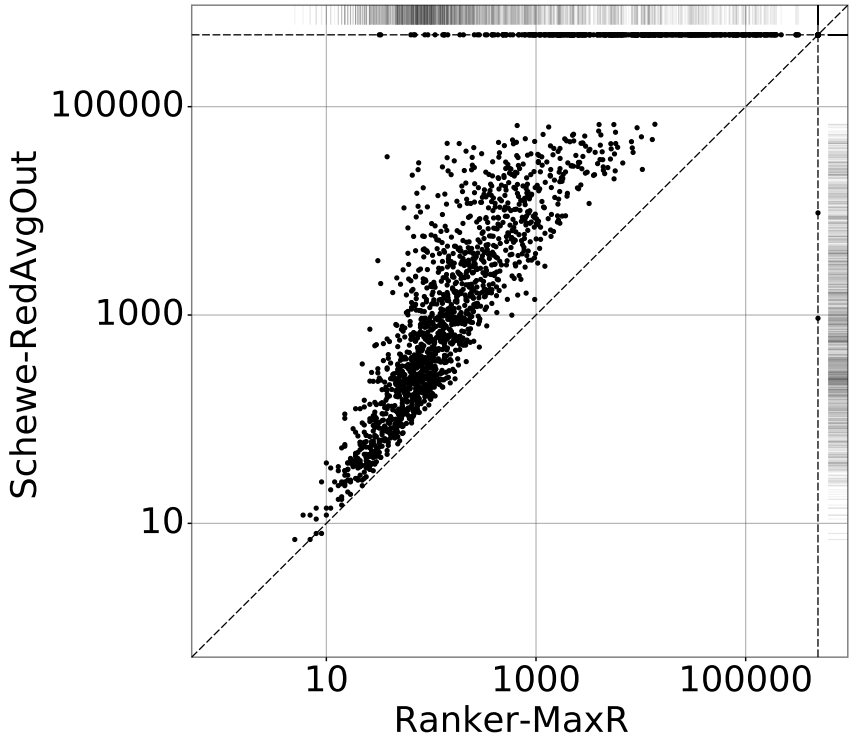
- Random automata $\Sigma = \{a, b\}$ from [Tsai,Fogarty,Vardi,Tsay'11]
  - starting with 15 states
  - reduced using SPOT, RABIT
  - removed semi-deterministic, inherently weak, unambigous, empty language
  - 2393 hard automata
  - Timeout: 5 min

Experimental Evaluation

- Random automata $\Sigma = \{a, b\}$ from [Tsai,Fogarty,Vardi,Tsay'11]
  - starting with 15 states
  - reduced using SPOT, RABIT
  - removed semi-deterministic, inherently weak, unambigous, empty language
  - 2393 hard automata
  - Timeout: 5 min

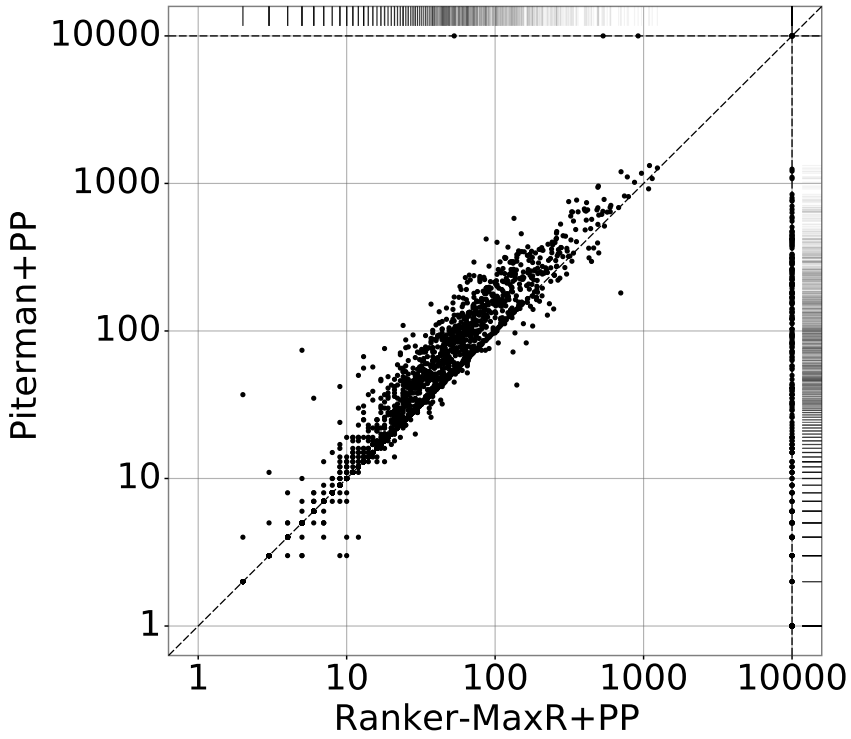
- Implemented in C++ within RANKER
  - RANKER+PIT: combination of RANKER and PITERMAN $\rightsquigarrow$ particular algorithm chosen according to WAITING and rank estimation
  - compared with:
    - GOAL⚽ (SCHEWE$_{\text{REDAVGOUT}}$, SAFRA, PITERMAN, FRIBOURG)
    - SPOT,
    - LTL2DSTAR,
    - SEMINATOR 2,
    - ROLL

Experimental Evaluation–States *rank-based*



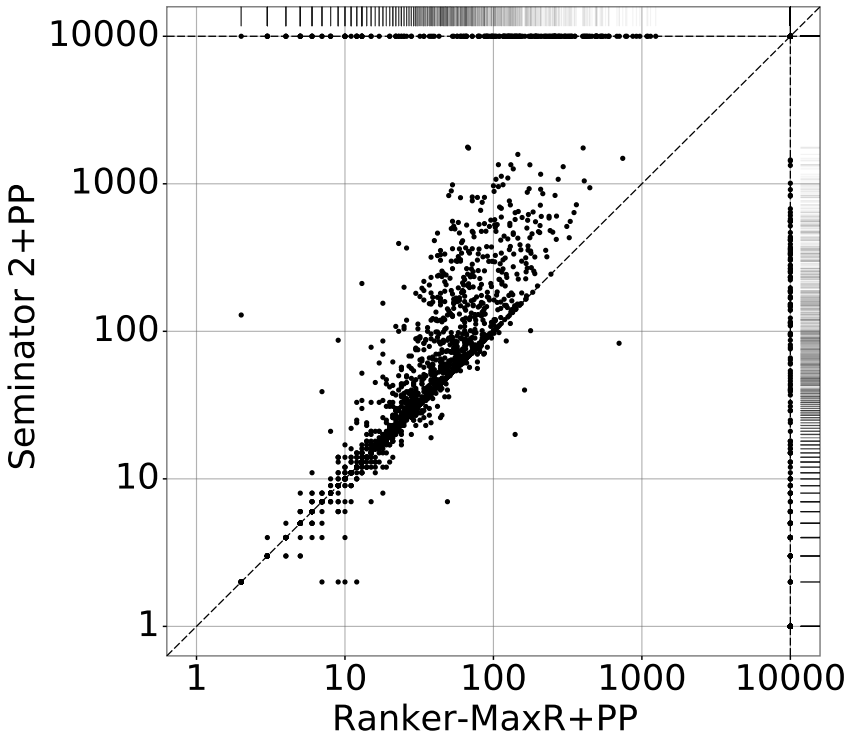
- SCHEWE$_{\text{REDAVGOUT}}$
- no postprocessing
- significant part of the state space pruned

Experimental Evaluation–States *not rank-based 1*

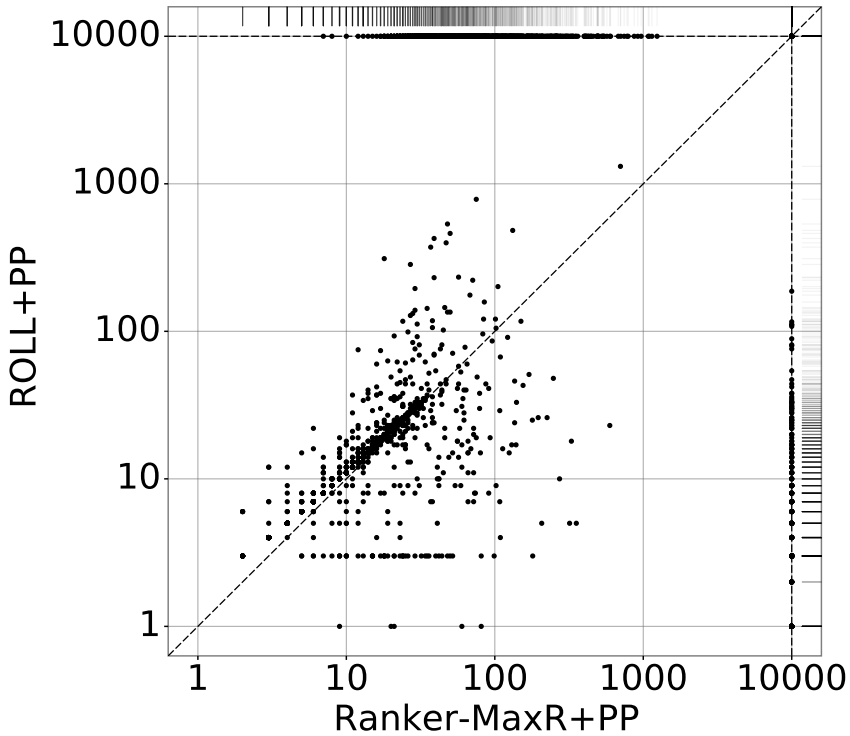


- PITERMAN (from GOAL ⚽)
  - on average better than SAFRA, SPOT, LTL2DSTAR
- determinization-based
- postprocessed by `autfilt`

Experimental Evaluation–States *not rank-based 2*



- SEMINATOR 2
- semideterminization-based
- postprocessed



- ROLL
- learning-based
- postprocessed

Experimental Evaluation–States *Cont.*

method	max	mean	med.	std. dev	TO	wins	(TO)	losses	(TO)
RANKER	319 119	8 051.58	185	28 891.4	360	—	—	—	—
SCHEWE$_{\text{REDAVGOUT}}$ ⚽	67 780	5 227.3	723	10 493.8	844	2030	(486)	3	(2)
RANKER	1 239	61.83	32	103.18	360	—	—	—	—
RANKER+PIT	1 706	73.65	33	126.8	17	—	—	—	—
PITERMAN ⚽	1 322	88.30	40	142.19	12	1 069	(3)	469	(351)
SAFRA ⚽	1 648	99.22	42	170.18	158	1 171	(117)	440	(319)
SPOT	2 028	91.95	38	158.13	13	907	(6)	585	(353)
FRIBOURG ⚽	2 779	113.03	36	221.91	78	996	(51)	472	(333)
LTL2DSTAR	1 850	88.76	41	144.09	128	1 156	(99)	475	(331)
SEMINATOR 2	1 772	98.63	33	191.56	345	1 081	(226)	428	(241)
ROLL	1 313	21.50	11	57.67	1 106	1 781	(1 041)	522	(295)

- RANKER: smallest mean and median (except ROLL), 15 % TO
- RANKER +PIT: 0.7 % TO
- In 22.5 % cases strictly smaller BA
- In 63.4 % cases at least as small as the best result

Conclusion

- Series of optimizations reducing the state space in rank-based complementation
- Competitive to other approaches, in 22.5 % cases strictly better

Conclusion

- Series of optimizations reducing the state space in rank-based complementation
- Competitive to other approaches, in 22.5 % cases strictly better

- Future work
  - more precise rank estimation according to automaton structure
  - generalization to TGBA
  - language inclusion checking

Conclusion

- Series of optimizations reducing the state space in rank-based complementation
- Competitive to other approaches, in 22.5 % cases strictly better

- Future work
  - more precise rank estimation according to automaton structure
  - generalization to TGBA
  - language inclusion checking

THANK YOU!

Experimental Evaluation–Time

method	mean	med.	std. dev
RANKER	10.21	0.84	28.43
RANKER+PIT	9.40	3.03	16.00
PITERMAN ⚽	7.47	6.03	8.46
SAFRA ⚽	15.49	7.03	35.59
SPOT	1.07	0.02	8.94
FRIBOURG ⚽	19.43	10.01	32.76
LTL2DSTAR	4.17	0.06	22.19
SEMINATOR 2	11.41	0.37	34.97
ROLL	42.63	14.92	67.31