Simulations in Rank-Based Büchi Automata Complementation

Yu-Fang Chen[1], Vojtěch Havlena[2], Ondřej Lengál[2]

[1] Academia Sinica, Taiwan

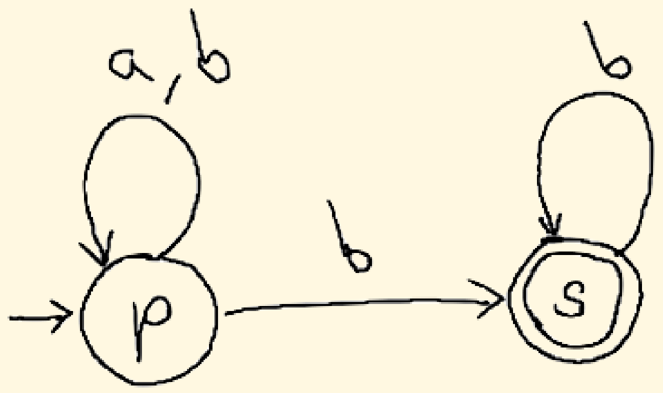
[2] Brno University of Technology, Czech Republic

Büchi Automata

- like Finite Automata, but on infinite words

$$\alpha = a_1 a_2 a_3 a_4 a_5 a_6 a_7 \dots \quad \in \Sigma^\omega$$

- $A = (Q, \Delta, Init, Acc)$ over Σ
  - Q – finite set of states
  - Δ – transition relation $\subseteq Q \times \Sigma \times Q$
  - $Init$ – initial states, Acc – accepting states
- a word is accepted by **looping** over an accepting state
- Büchi Automata recognize ω-regular languages
- language $L_{example} = (a+b)^* b^\omega$

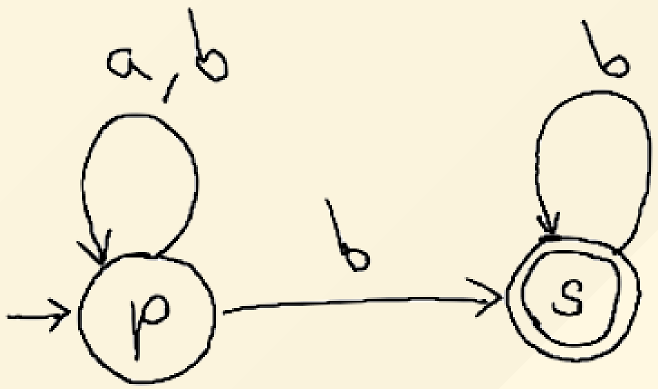


Büchi Automata — Motivation

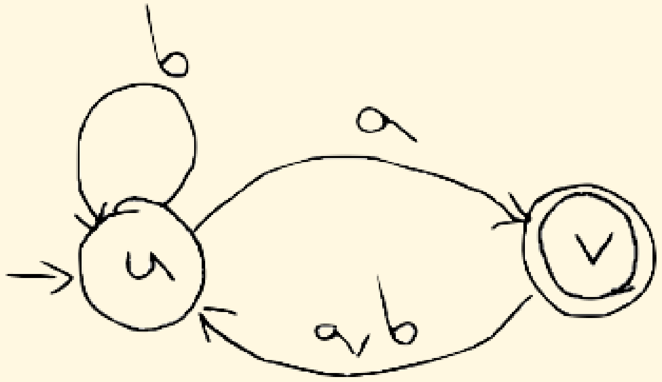
- model checking of linear-time properties (Vardi & Wolper)
  - property $\varphi \leadsto A_\varphi$
  - system $S \leadsto A_S$
  - checking $S \models \varphi \quad \leadsto \quad A_S \subseteq A_\varphi \quad \leadsto \quad A_S \cap A_\varphi^C = \emptyset$
- termination analysis (e.g. **Ultimate Automizer**: Heizmann, Hoenicke, Podelski)
- decision procedure for S1S (Büchi)
  - monadic 2^{nd} order logic over $(\mathbb{N}, 0, +1)$

Complementing Büchi Automaton

- $L(A^C) = \Sigma^\omega \setminus L(A)$
- much more involved than for Finite Automata (cannot be determinized)
- example
  - $L(A_1) = (a + b)^* b^\omega$
  - $\overline{L(A_1)} = (b^* a)^\omega$



$\rightsquigarrow$



Complementing Büchi Automaton

Why need to complement Büchi Automata?

- **Termination Analysis** – Ultimate Automizer (Heizmann, Hoenicke, Podelski)
  - A_{ToDo} – Büchi Automaton representing a set of uprocessed program traces
  - **while** $L(A_{ToDo}) \neq \emptyset$:
    1. pick a word $\alpha \in L(A_{ToDo})$ and prove its termination
    2. generalize α to a set of words with the same termination argument: A_α
    3. $A_{ToDo} := \quad A_{ToDo} \setminus A_\alpha \quad = \quad A_{ToDo} \cap A_\alpha^C$
- **Decision Procedures** of logics (negation):
  - S1S: monadic 2^{nd} order logic over $(\mathbb{N}, 0, +1)$
  - ETL: extended temporal logic
  - QPTL: quantified propositional temporal logic

Complementing Büchi Automaton

- Büchi's original construction (1962) $A \rightsquigarrow A^C$:
  - based on infinite Ramsey theorem
  - size: $2^{2^{O(n)}}$ n – number of states of A
- Safra's algorithm (1988):
  - through deterministic Rabin automaton
  - size: $2^{\mathcal{O}(n \cdot \log n)}$
- Ramsey-based [Sistla, Vardi, Volper '87]
- determinization-based [Safra '88], [Piterman '06]
- **rank-based** [Kupferman, Vardi '01], [Schewe '09]
- slice-based [Kähler, Wilke '08], [Vardi, Wilke '08]
- learning-based [Li, Turrini, Zhang, Schewe '18]
- subset-tuple construction [Allred, Ultes-Nitsche '18]

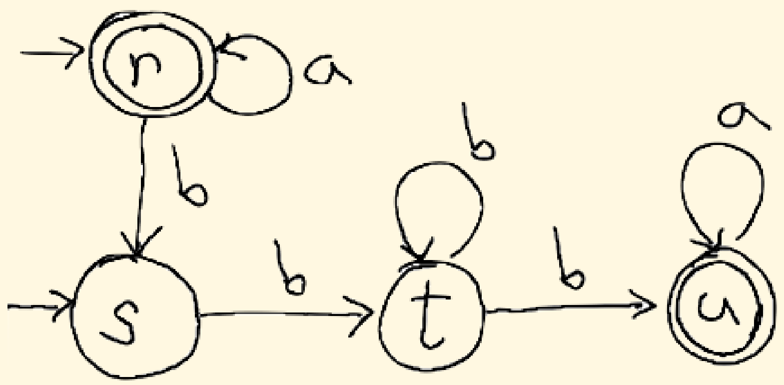
Our Contribution

- improvement of the **rank-based** Büchi Automata complementation procedure
  - started by [Kupferman, Vardi '01]
  - several optimizations
  - [Schewe '09] – **complexity-tight** size: $(0.76)^n$ (modulo $\mathcal{O}(n^2)$)
- we use **simulations** for two optimizations
  1. **purging** macrostates with simulation-incompatible rankings (always helps)
  2. **saturating** macrostates (can help merge several states)
- in some practical settings, the optimizations are **for free**

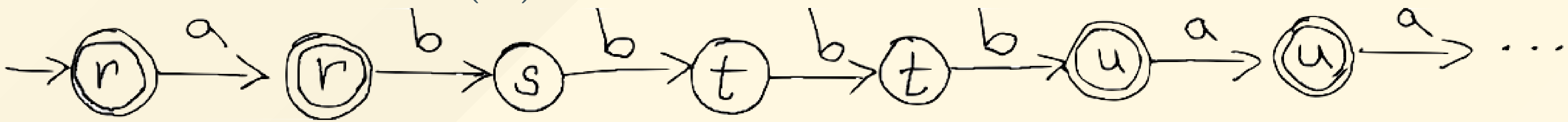
Rank-based Büchi Automaton Complement

- we will show the ideas of our contribution on the original [Kupferman, Vardi '01]
- easy extension to the optimal [Schewe '09]

$$L(A) = a^*(a^\omega + bbb^+a^\omega) + bb^+a^\omega$$



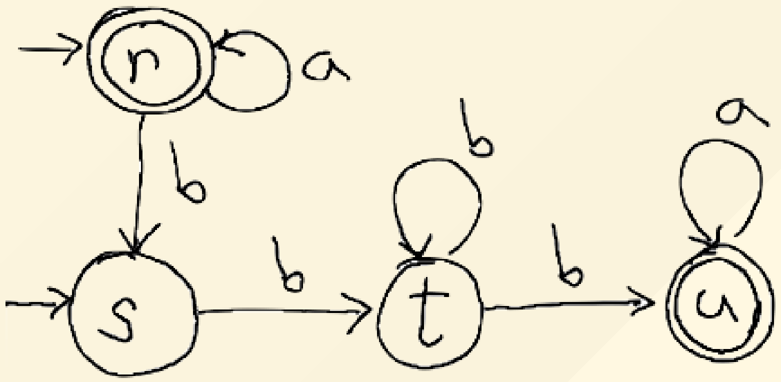
$$\alpha = abbbbaa\ldots a^\omega \in L(A)$$

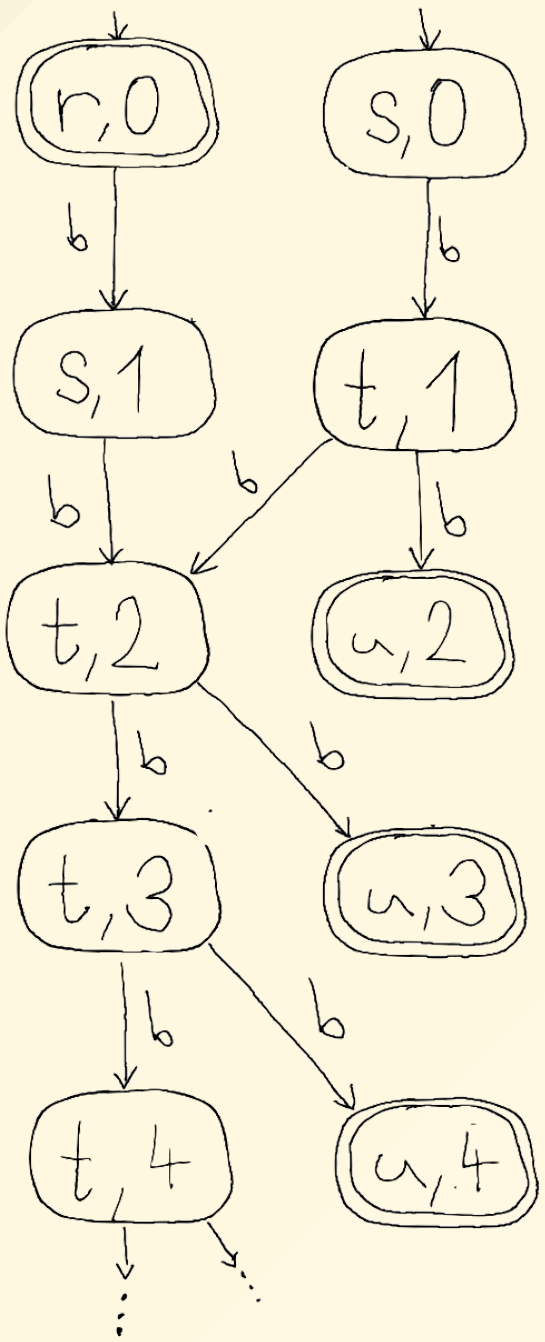


Rank-based BA^C — Run DAG

Run DAG G_α of A on $\alpha \in \Sigma^\omega$

- represents all runs of A on α
- nodes are $(state, step)$
  - $state \in Q$, $step \in \mathbb{N}$
- example: $\alpha = b^\omega \notin L(A)$





Rank-based BA^C — Run DAG

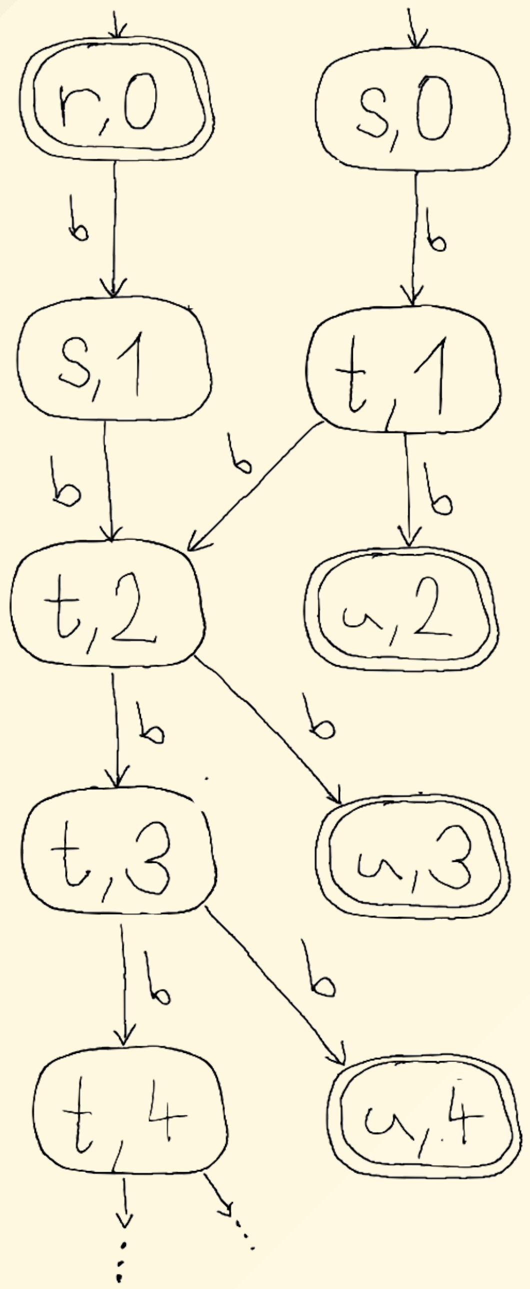
- assigns **ranks** to nodes of Run DAG G_α

$i := 0$

while $i \leq 2 \cdot |Q|$:

1. assign rank i to nodes with finitely many successors and remove them from G_α
2. assign rank $i + 1$ to nodes that cannot reach Acc and remove them from G_α
3. $i := i + 2$

Lemma: [Kupferman, Vardi '01]
If $\alpha \notin L(A)$, then $\forall n \in G_\alpha : rank(n) \leq 2 \cdot |Q|$.



Rank-based BA^C — Run DAG

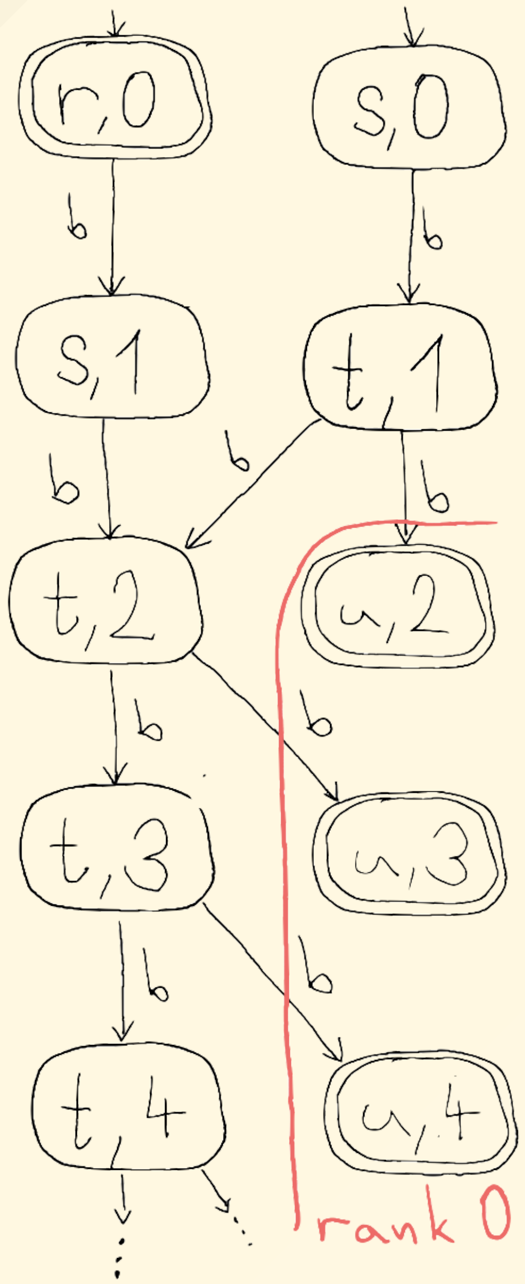
- assigns **ranks** to nodes of Run DAG G_α

  $i := 0$

  `while` $i \leq 2 \cdot |Q|$:

  1. assign rank i to nodes with finitely many successors and remove them from G_α
  2. assign rank $i + 1$ to nodes that cannot reach Acc and remove them from G_α
  3. $i := i + 2$

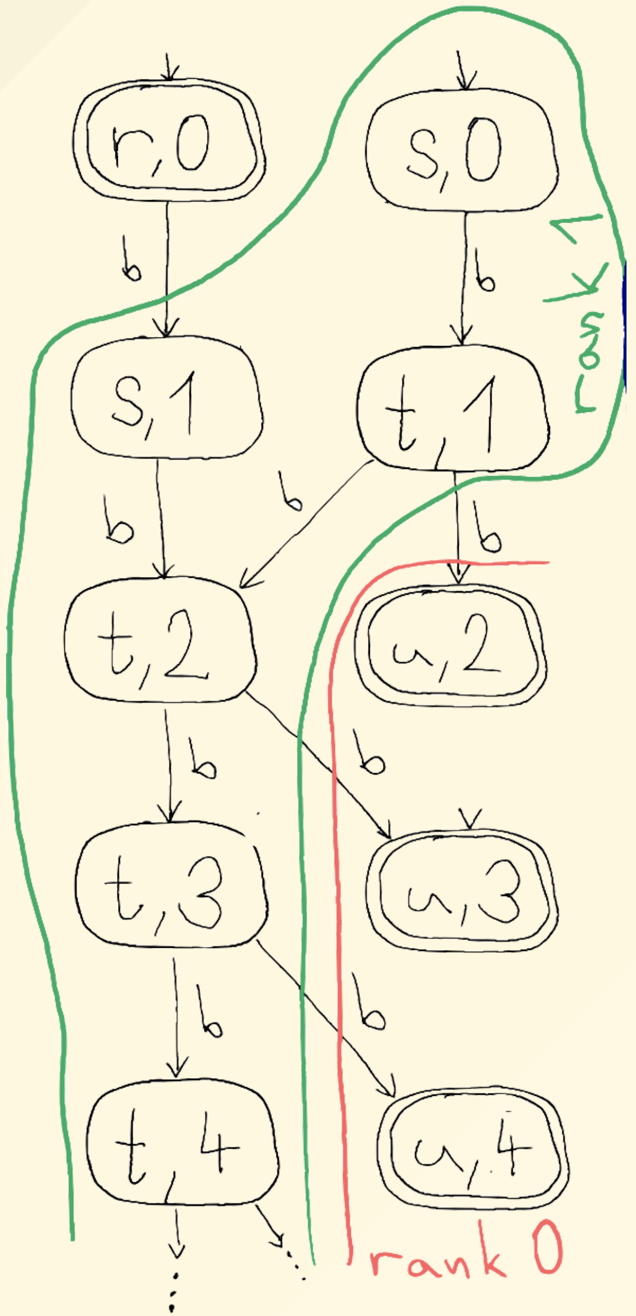
Lemma: [Kupferman, Vardi '01]
If $\alpha \notin L(A)$, then $\forall n \in G_\alpha : rank(n) \leq 2 \cdot |Q|$.



Rank-based BA^C — Run DAG

- assigns **ranks** to nodes of Run DAG G_α
  $i := 0$
  while $i \leq 2 \cdot |Q|$:
  1. assign rank i to nodes with finitely many successors and remove them from G_α
  2. assign rank $i + 1$ to nodes that cannot reach Acc and remove them from G_α
  3. $i := i + 2$

Lemma: [Kupferman, Vardi '01]
If $\alpha \notin L(A)$, then $\forall n \in G_\alpha : rank(n) \leq 2 \cdot |Q|$.

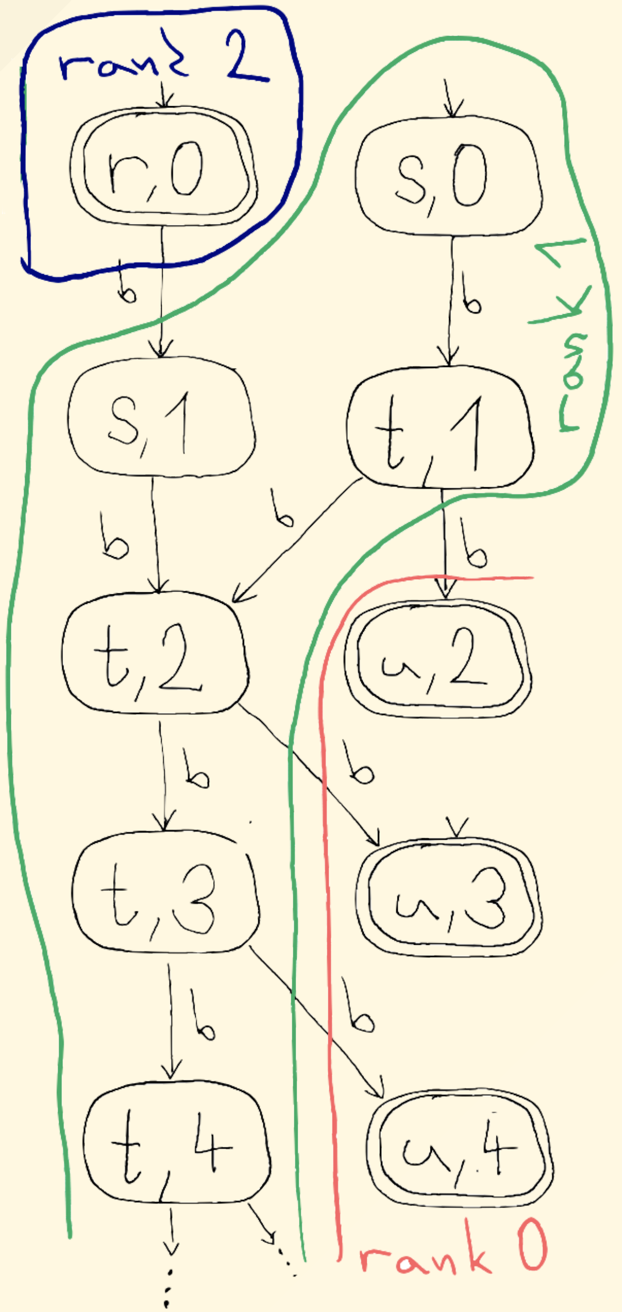


Rank-based BA^C — Run DAG

- assigns **ranks** to nodes of Run DAG G_α
  $i := 0$
  while $i \leq 2 \cdot |Q|$:

  1. assign rank i to nodes with finitely many successors and remove them from G_α
  2. assign rank $i + 1$ to nodes that cannot reach Acc and remove them from G_α
  3. $i := i + 2$

Lemma: [Kupferman, Vardi '01]
If $\alpha \notin L(A)$, then $\forall n \in G_\alpha : rank(n) \leq 2 \cdot |Q|$.

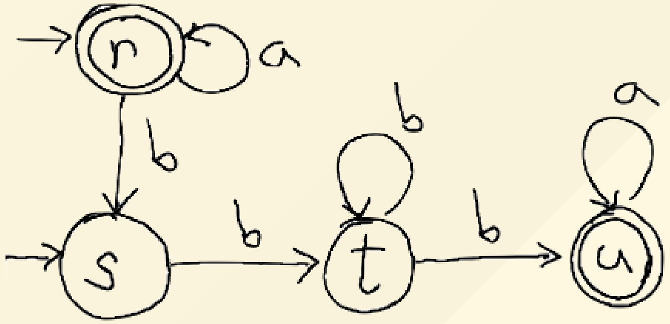


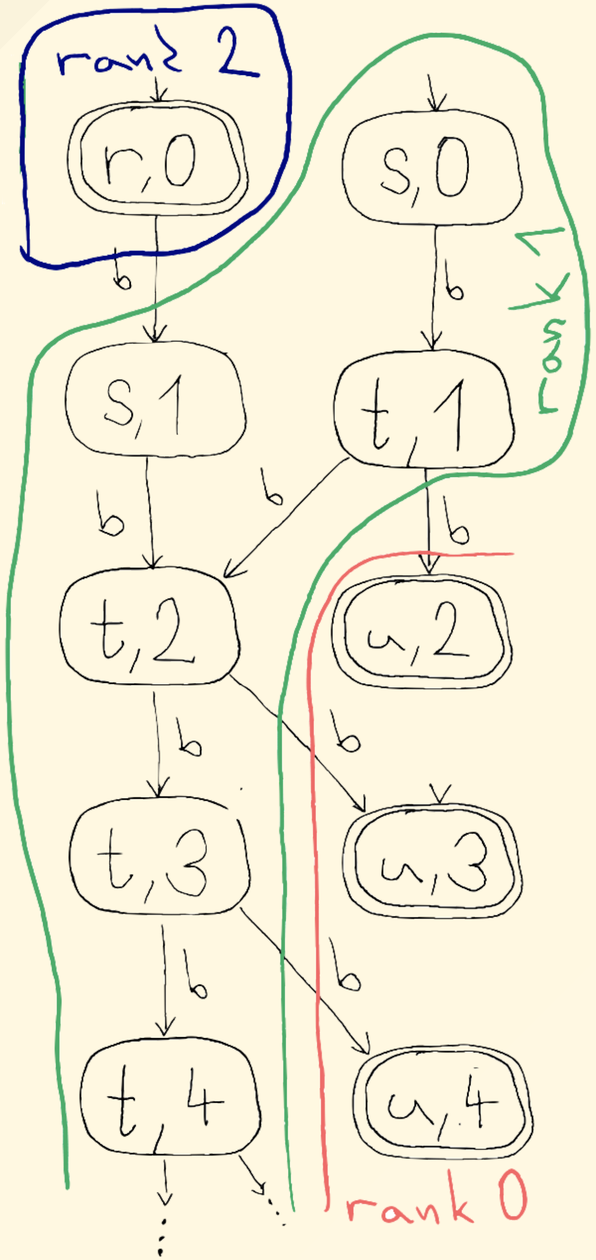
Rank-based BA^C

- $A^C = (Q^C, \Delta^C, \quad Init^C = \{Init\} \times \{\emptyset\} \times ?, \qquad Acc^C = Q^C \times \{\emptyset\} \times ?)$
  - $Q^C = \{(det, cut, rank) \mid det, cut \in 2^Q, rank : Q \to \{0, \ldots, 2n\}\}$
    - det: states reachable in runs over the same word
    - cut: represents runs that need to leave Acc
    - $rank$: a guess of ranking of Run DAG nodes
  - Δ^C : we have $(det, cut, rank) \xrightarrow{\{a\}} (det', cut', rank') \in \Delta^C$
    - $det' = Post_a(det)$
    - $rank'$: non-incr wrt Δ from $rank$ s.t. $rank'(q_{Acc})$ is even for $q_{Acc} \in Acc$
    - $cut' = Post_a(cut) \setminus odd(rank')$ if $cut \neq \emptyset$
      $Post_a(det) \setminus odd(rank')$ otherwise

Rank-based BA^C

Example: $b^\omega \in L(A^C)$





0. $(\{\mathbf{r}, \mathbf{s}\}, \emptyset, \{\mathbf{r} \mapsto \mathbf{2}, \mathbf{s} \mapsto \mathbf{1}\})$

$$\downarrow b$$

1. $(\{\mathbf{s}, \mathbf{t}\}, \emptyset, \{\mathbf{s} \mapsto \mathbf{1}, \mathbf{t} \mapsto \mathbf{1}\})$

$$\downarrow b$$

2. $(\{t, u\}, \{u\}, \{t \mapsto 1, u \mapsto 0\})$

$$\downarrow b$$

3. $(\{\mathbf{t}, \mathbf{u}\}, \emptyset, \{\mathbf{t} \mapsto \mathbf{1}, \mathbf{u} \mapsto \mathbf{0}\})$

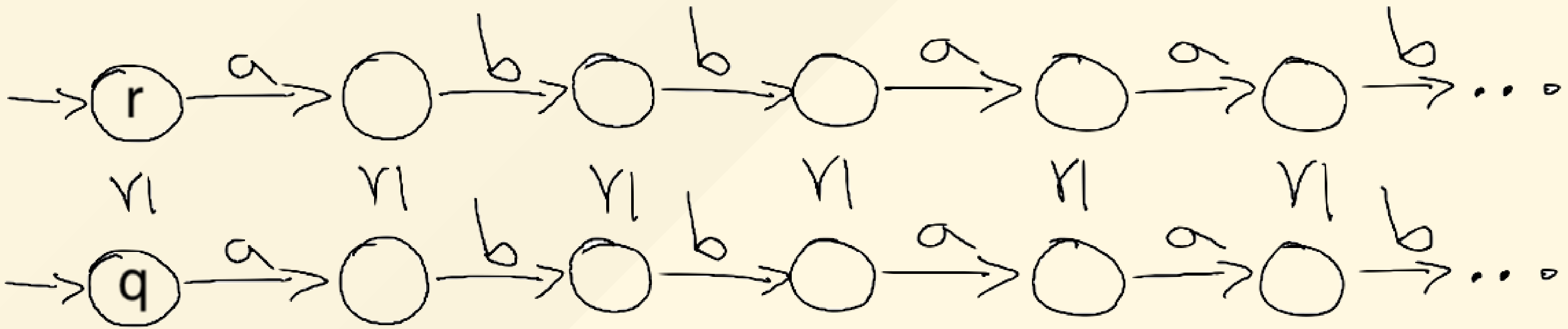
$$\downarrow b$$

4. $(\{t, u\}, \{u\}, \{t \mapsto 1, u \mapsto 0\})$ (2)

$$\vdots$$

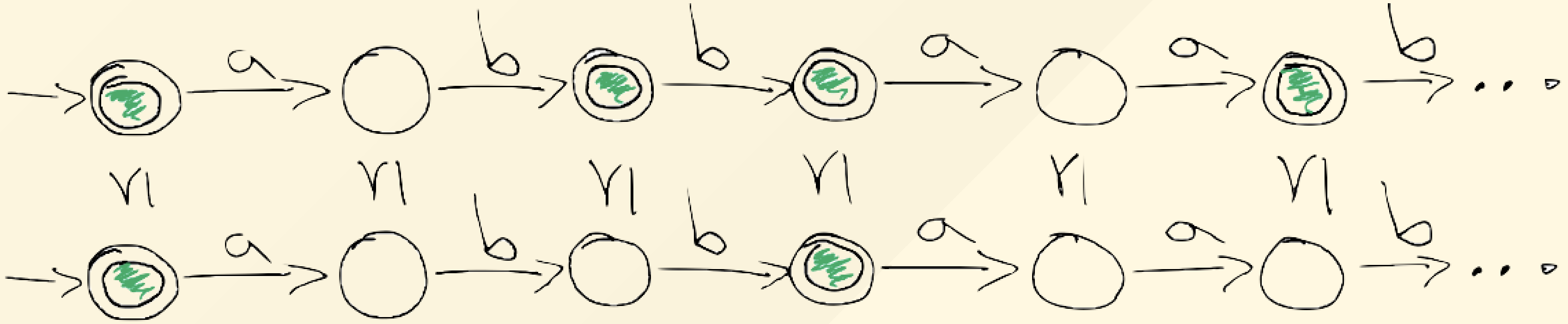
Simulations

- often used to speed-up expensive FA & BA operations (inclusion, reduction)
- Let $A = (Q, \Delta, Init, Acc)$
- relation on states $\preceq \subseteq Q \times Q$
- $q \preceq r \quad \Rightarrow \quad L(q) \subseteq L(r)$
- generally: if $q \overset{\{a\}}{\longrightarrow} q'$, then it needs to hold that $r \overset{\{a\}}{\longrightarrow} r'$ where $q' \preceq r'$

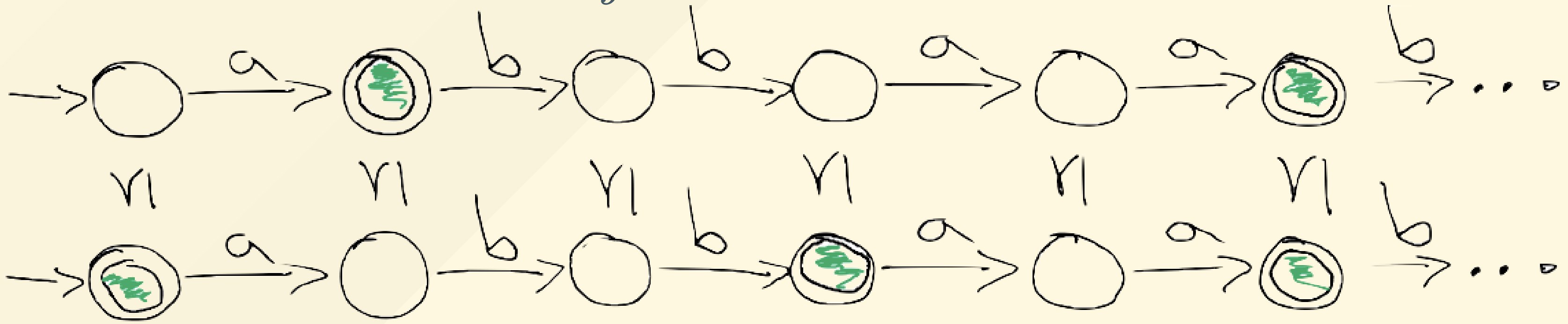


Simulations

- various relations based on handling Acc:
  - **direct** simulation $\preceq_{direct}$



  - **delayed** simulation $\preceq_{delayed}$ (note that $\preceq_{direct} \subseteq \preceq_{delayed}$)



- both can be used for quotienting (merging states p and q s.t. $p \preceq q \wedge q \preceq p$)

Our Contribution #1 — Purging

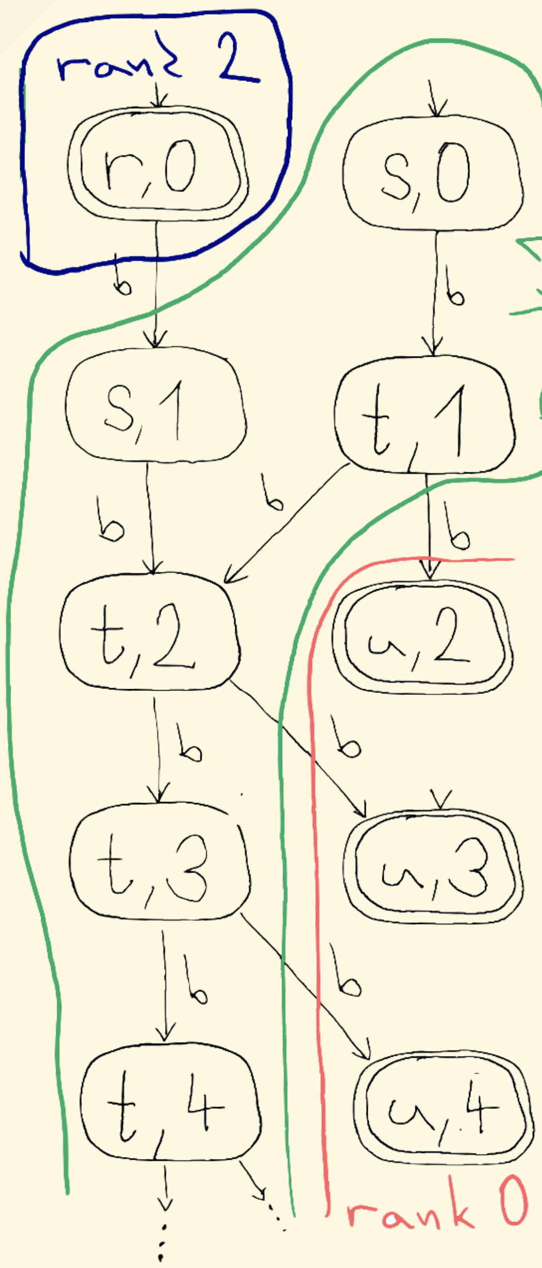
Modification of the rank-based complementation.

Idea: if $q \preceq r$ and G_α is the Run DAG of A on $\alpha \in \Sigma^\omega$, then the subgraph of G_α rooted at a q-node is embedded in a subgraph rooted in a r-node on the same level.

Therefore, the rank of a q-node will never be higher* than the rank of an r-node on the same level.

Theorem 1: Let $A = (Q, \Delta, Init, Acc)$ be a BA such that $q \preceq_{direct} r$.
Then we can remove from A^C all $(det, cut, rank)$ where

$$q, r \in det \qquad \text{and} \qquad rank(q) > rank(r)$$



Our Contribution #1 — Purging

Modification of the rank-based complementation.

Theorem 2: Let $A = (Q, \Delta, Init, Acc)$ be a BA such that $q \preceq_{delayed} r$.
Then we can remove from A^C all macrostates $(det, cut, rank)$ where

$$q, r \in det \qquad \text{and} \qquad rank(q) > \llbracket rank(r) \rrbracket$$

where $\llbracket rank(r) \rrbracket$ is the smallest even number $\geq rank(r)$

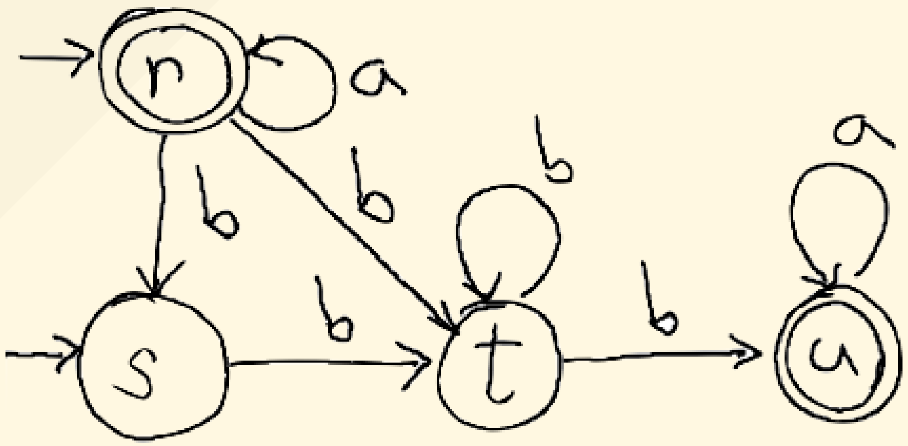
Theorem 3: (combines **Theorem 1** and **Theorem 2**)
We can remove from A^C all macrostates $(det, cut, rank)$ where $q, r \in det$ and

$$q \preceq_{direct} r \qquad \text{and} \qquad rank(q) > rank(r) \qquad \text{or}$$

$$q \preceq_{delayed} r \qquad \text{and} \qquad rank(q) > \llbracket rank(r) \rrbracket$$

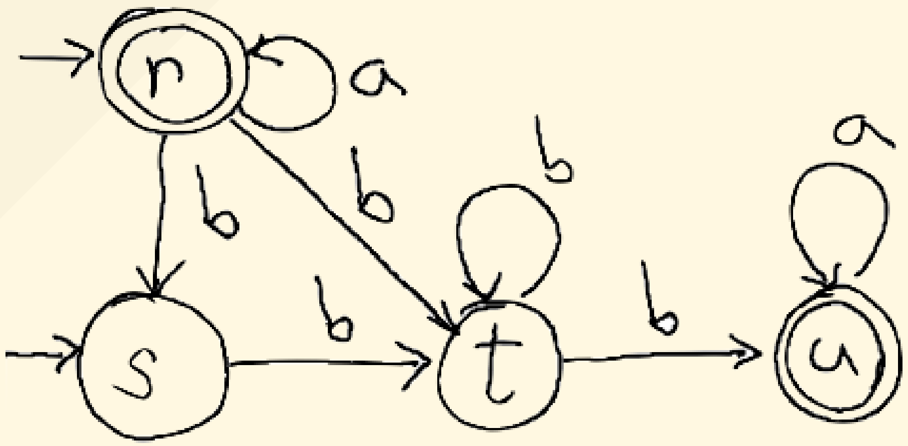
Our Contribution #1 — Purging

- $s \preceq_{delayed} r$
- initial states of A^C:
  - $(\{r, s\}, \emptyset, \{r \mapsto 2, s \mapsto 1\})$
  - $(\{r, s\}, \emptyset, \{r \mapsto 2, s \mapsto 2\})$
  - $(\{r, s\}, \emptyset, \{r \mapsto 2, s \mapsto 3\})$
  - $(\{r, s\}, \emptyset, \{r \mapsto 2, s \mapsto 4\})$
  - $(\{r, s\}, \emptyset, \{r \mapsto 2, s \mapsto 5\})$
  - ...



Our Contribution #1 — Purging

- $s \preceq_{delayed} r$
- initial states of A^C:
  - $(\{r, s\}, \emptyset, \{r \mapsto 2, s \mapsto 1\})$
  - $(\{r, s\}, \emptyset, \{r \mapsto 2, s \mapsto 2\})$
  - $(\{r, s\}, \emptyset, \{r \mapsto 2, s \mapsto 3\})$ XXX
  - $(\{r, s\}, \emptyset, \{r \mapsto 2, s \mapsto 4\})$ XXX
  - $(\{r, s\}, \emptyset, \{r \mapsto 2, s \mapsto 5\})$ XXX
  - ...
- **Always works** (never increases the size of A^C)



Our Contribution #2 — Saturation

Idea: Adding simulation-smaller states to det doesn't change its language.

$$q \preceq r \quad \Rightarrow \quad L(\{r\}) = L(\{r, q\})$$

Theorem 4: Let $A = (Q, \Delta, Init, Acc)$ be a BA.
Every macrostate $(det, cut, rank)$ in A^C can be changed to $(\mathbf{cl}[det], cut, rank')$,
where $\mathbf{cl}[det] = \{q \in Q \mid s \in det : q \preceq_{delayed} s\}$

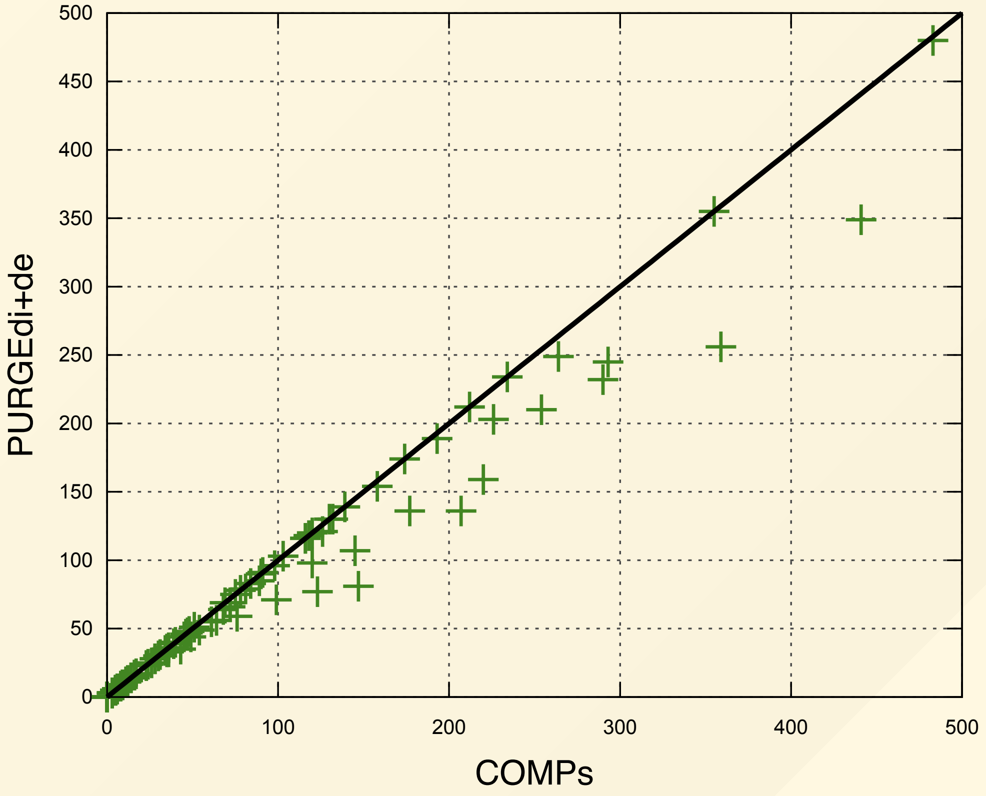
Theorem 4 can help us map $(det_1, cut_1, rank_1)$ and $(det_2, cut_2, rank_2)$ to the same macrostate $(det', cut', rank')$.

- sometimes helps
- sometimes increases the size

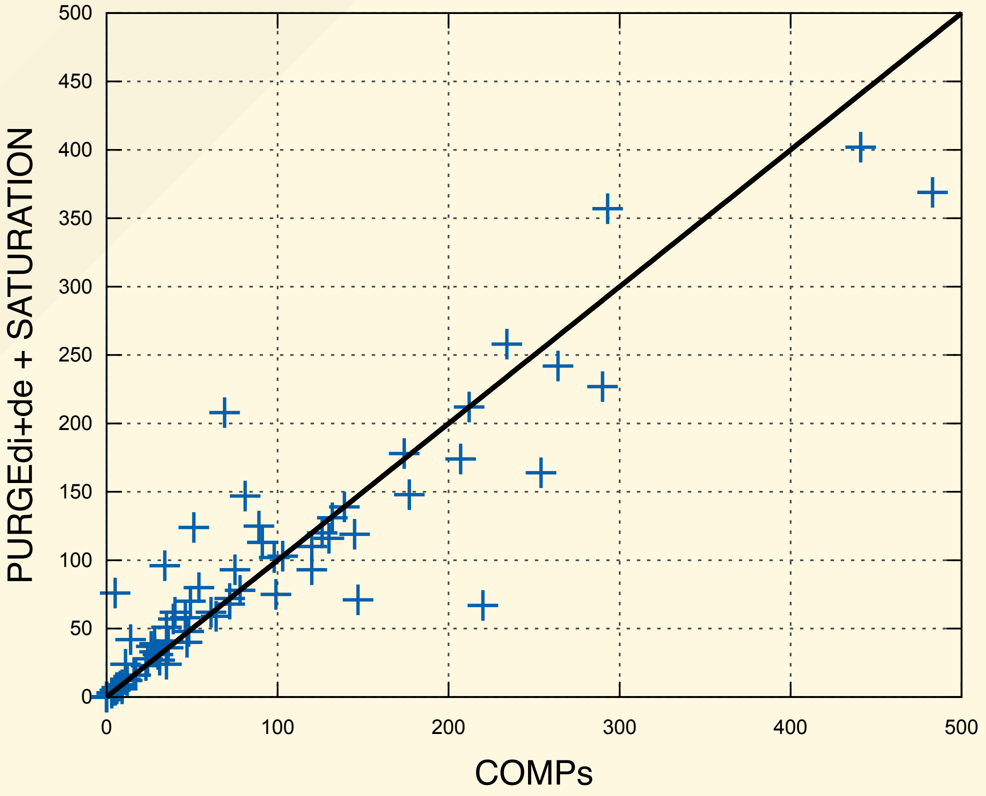
Experiments

- 124 random BAs (w/ non-trivial language) over $\Sigma = \{a, b\}$
  - quotiented wrt delayed simulation

Experiments



a) PURGE vs. original



b) PURGE+SATURATION vs. original

- Best result: from 4065 to 985 (PURGE only) to 929 (PURGE + SATURATE)

Conclusion

- use of simulation to **optimize** complement of Büchi Automata
  - **purging**: remove macrostates with simulation-incompatible ranking
  - **saturation**: saturate macrostates, maybe some will merge
- the optimization is in some practical settings **for free**

Future work

- extend to other complement constructions
- extend to richer simulations
  - multi-pebble
  - look-ahead