Divide-and-Conquer Determinization of Büchi automata Based on SCC Decomposition

CAV Artifact Evaluation ★ Available

Yong Li, Andrea Turrini, Weizhi Feng, Moshe Y. Vardi and Lijun Zhang

ISCAS 中国科学院软件研究所 Institute of Software Chinese Academy of Sciences

RICE UNIVERSITY

CAV Artifact Evaluation ★★★ Reusable

Büchi determinization

Nondeterministic Büchi automata (**NBA**)

↓

Deterministic ω-automata with **more general** conditions:

- **Rabin** condition (**DRA**)
- **Parity** condition (**DPA**)
- **Emerson-Lei** condition (**DELA**) (**this work**)

Büchi determinization

Nondeterministic Büchi automata (**NBA**)

Deterministic ω-automata with **more general** conditions:

- **Rabin** condition (**DRA**)
- **Parity** condition (**DPA**)
- **Emerson-Lei** condition (**DELA**) (**this work**)

Büchi automata are not closed under determinization

Why Büchi determinization is important

- **Reactive synthesis**
- **Probabilistic verification**
- **Complementing Büchi automata**
- **Checking language inclusion**
  - **Pecan** theorem prover via **Spot**

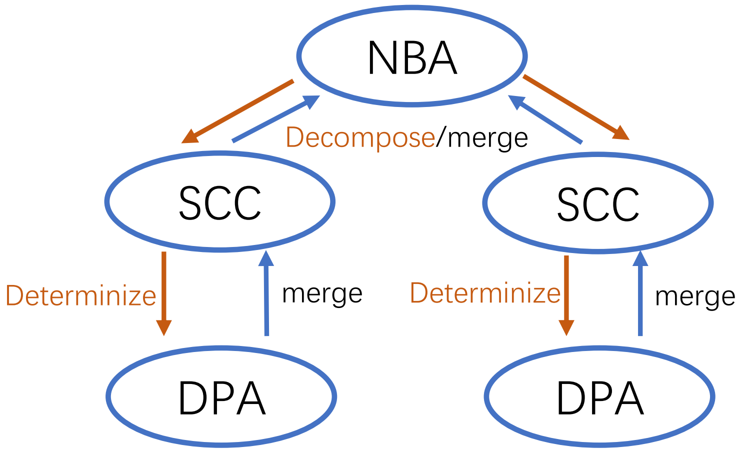
Büchi determinization is hard

- **NFA** determinization
  - Subset construction, 2^n
- **NBA** determinization
  - Subset construction + two preorders
  - Complexity: $O((n!)^2) \in 2^{O(n\log n)}$
  - Safra-Piterman's tree

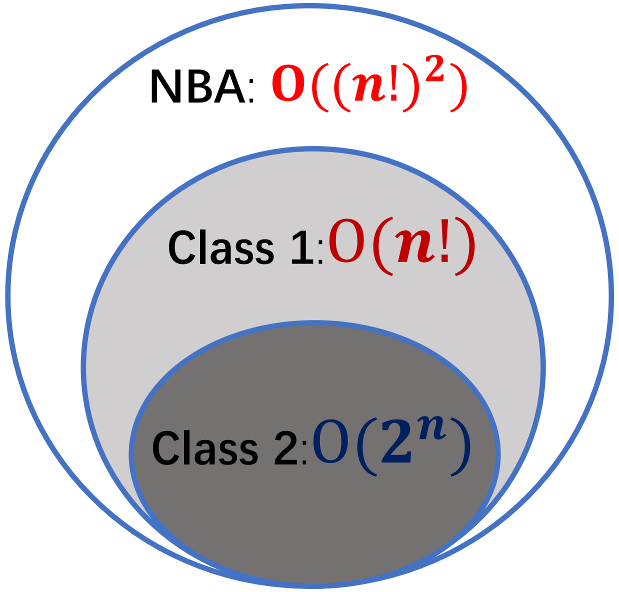
Work on automaton graph in whole

Our contributions

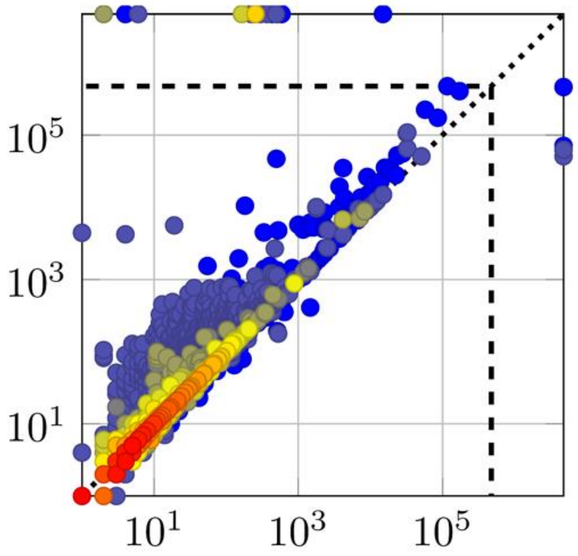
1. Divide-and-conquer methodology



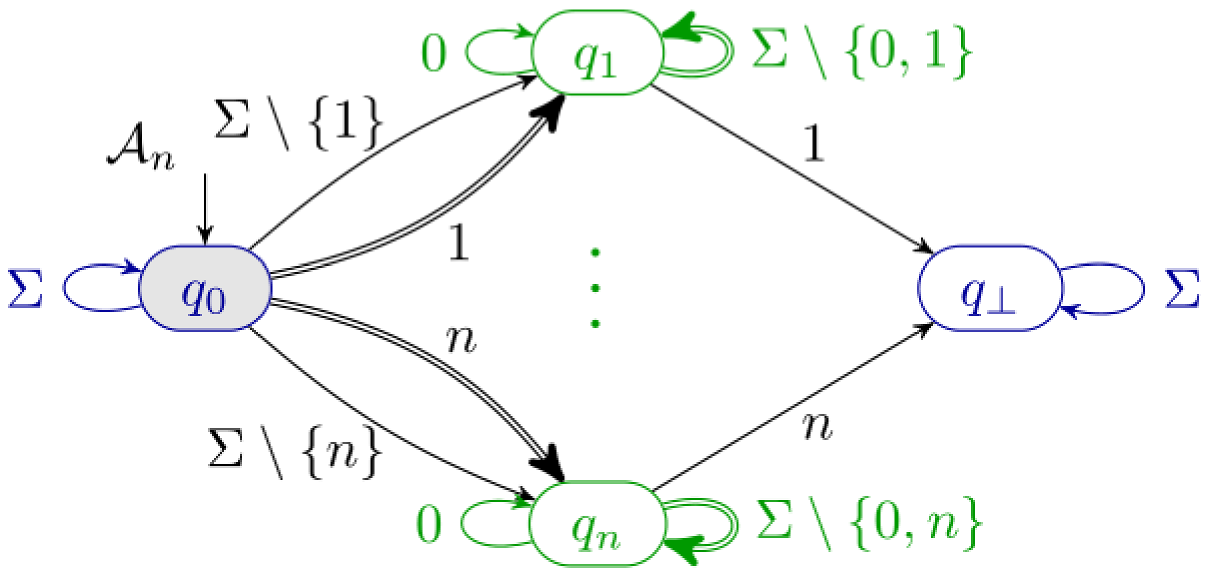
2. Two subclasses with better upper bounds



3. Comprehensive evaluation



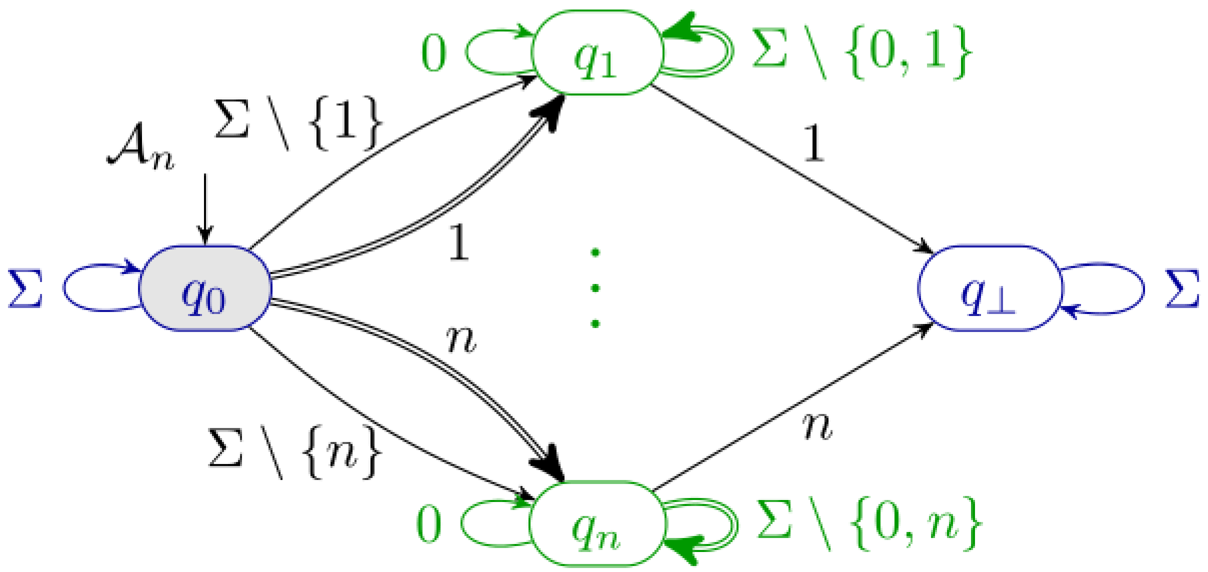
Insights in NBA determinization



Spot/Owl: n! states

Need all possible orders
over $\{q_1, q_2, \cdots, q_n\}$

Insights in NBA determinization



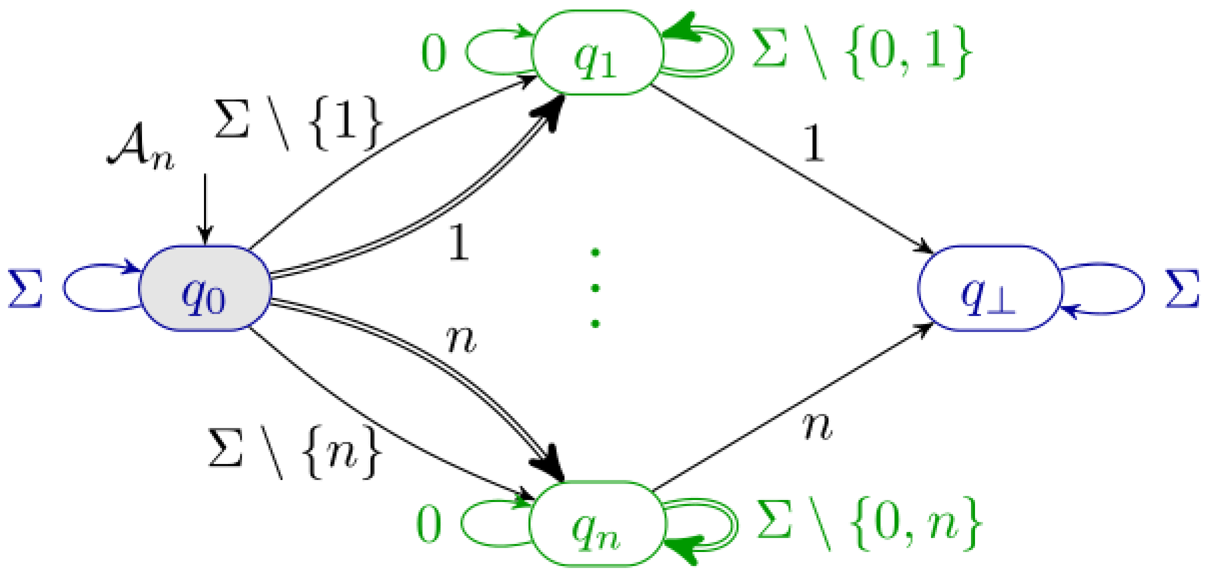
q_i and q_j **can not** reach each other

No need to put **preorder on all of** q_i-states

Insight 1:

Determinize each SCC independently

Divide-and-Conquer determinization



Divide-and-Conquer:
preorders for **each** SCC **independently**

Runs in different SCCs will not affect each other

Insights in Büchi automata

Three different types of SCCs

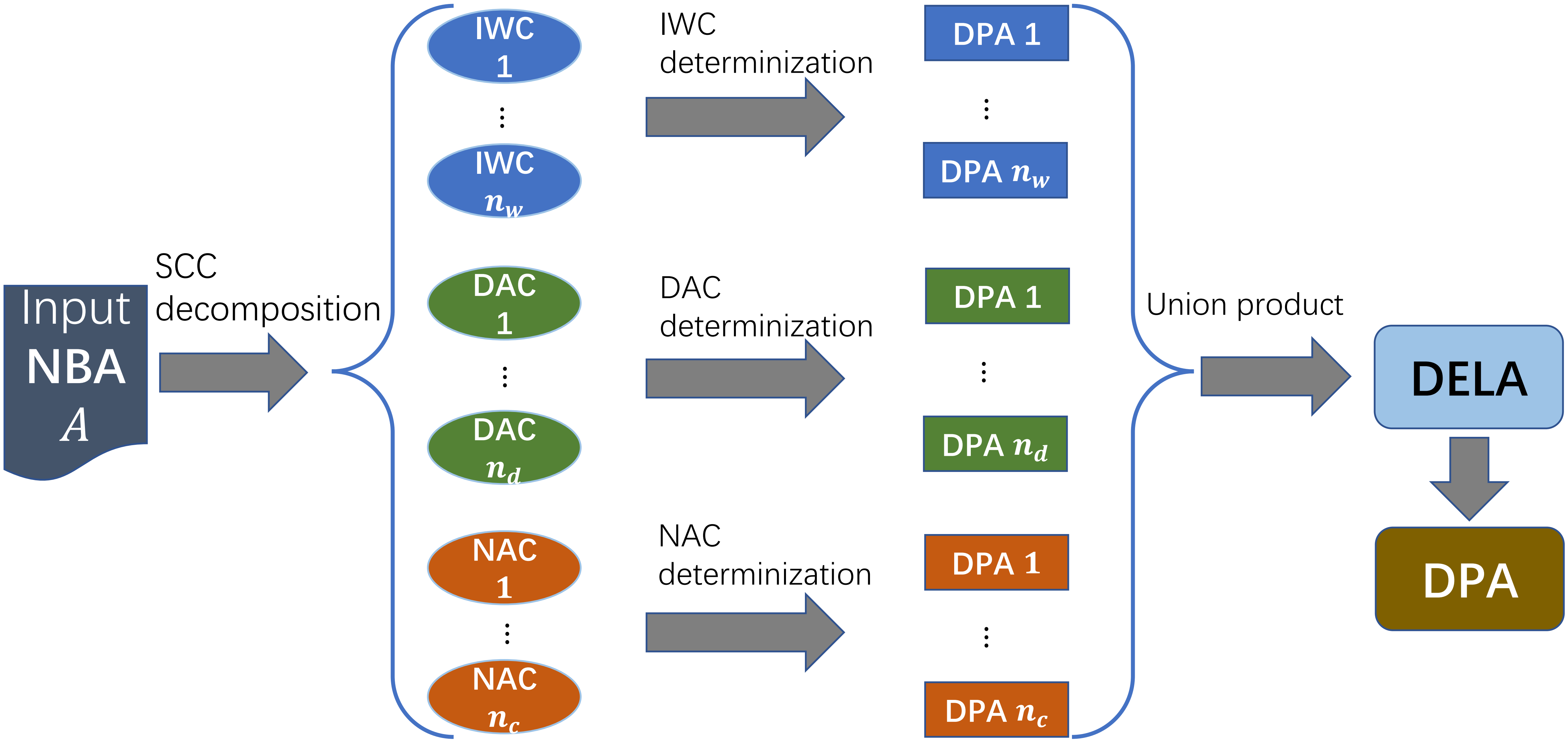
1. **Inherently Weak SCC (IWC):**
   - All cycles are either accepting or rejecting
2. **Deterministic Accepting SCC (DAC):**
   - Deterministic inside SCC
3. **Nondeterministic Accepting SCC (NAC):**
   - Remaining SCCs

Determinizing different types of SCCs

1. Inherently Weak SCC (IWC): 3^n
2. Deterministic Accepting SCC (DAC): $O(n!)$
3. Nondeterministic Accepting SCC (NAC): $O((n!)^2)$

Insight 2:
Specific construction for each type of SCC

Our determinization construction



Perform union product on-the-fly

Main results

Complexity:

1. **General** Büchi automata: $\mathbf{O((n!)^2)}$ **--same state of art**

2. **Weak** Büchi automata (with only **IWCs**): $\mathbf{3^n}$ **--same state of art**

3. **Better upper bounds for two subclasses**:
   - NBA with only **IWCs** and **DACs**: $\mathbf{O(n!)}$ vs. $\mathbf{O((n!)^2)}$
   - NBA with **one IWC** and **DACs** with **one sink state** : $\mathbf{O(2^n)}$ vs. $\mathbf{O(n!)}$

Empirical evaluation

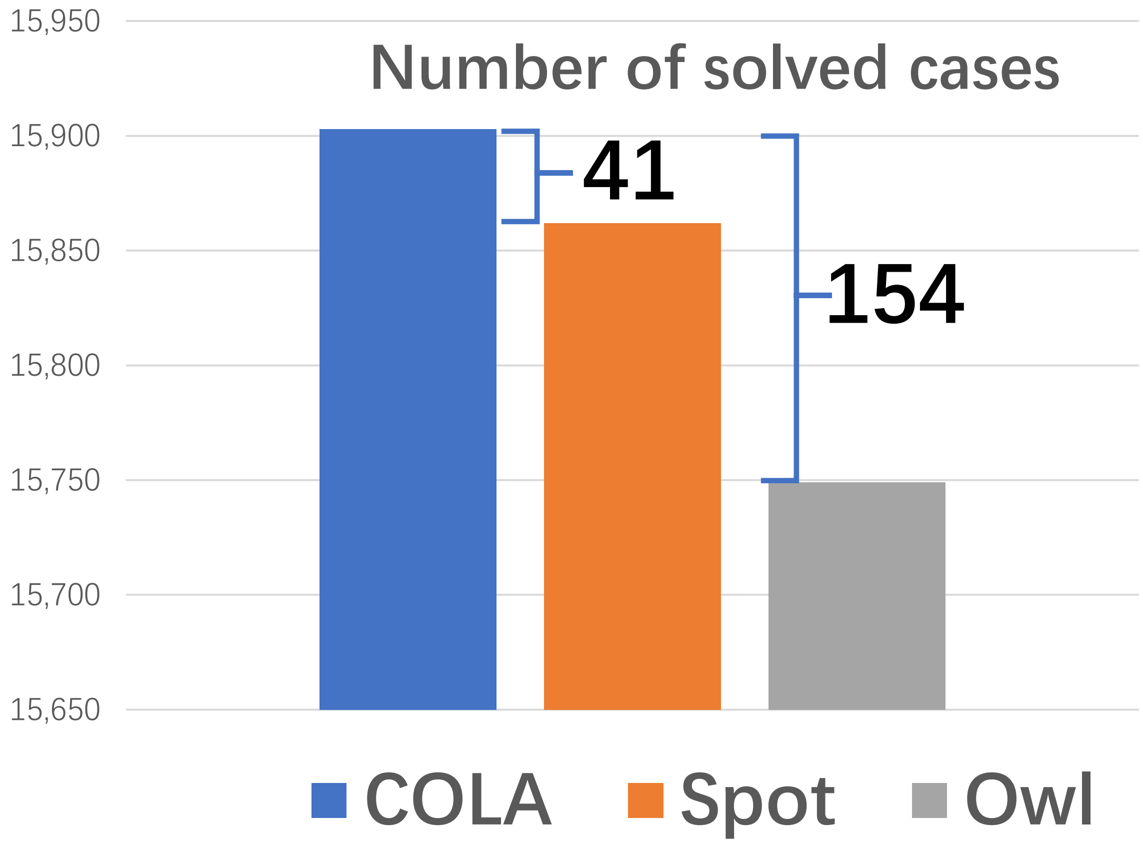
- **COLA** built on top of **Spot**
  - Our divide-and-conquer construction
- **Spot**
  - Safra-Piterman's approach
- **Owl**
  - Specific constructions for **IWCs** and **DACs**

- **Benchmark** set
  - **15,913** automata from literature
  - Output deterministic Parity automata

- **Comparison**
  - Runtime
  - Size of automata

Empirical evaluation

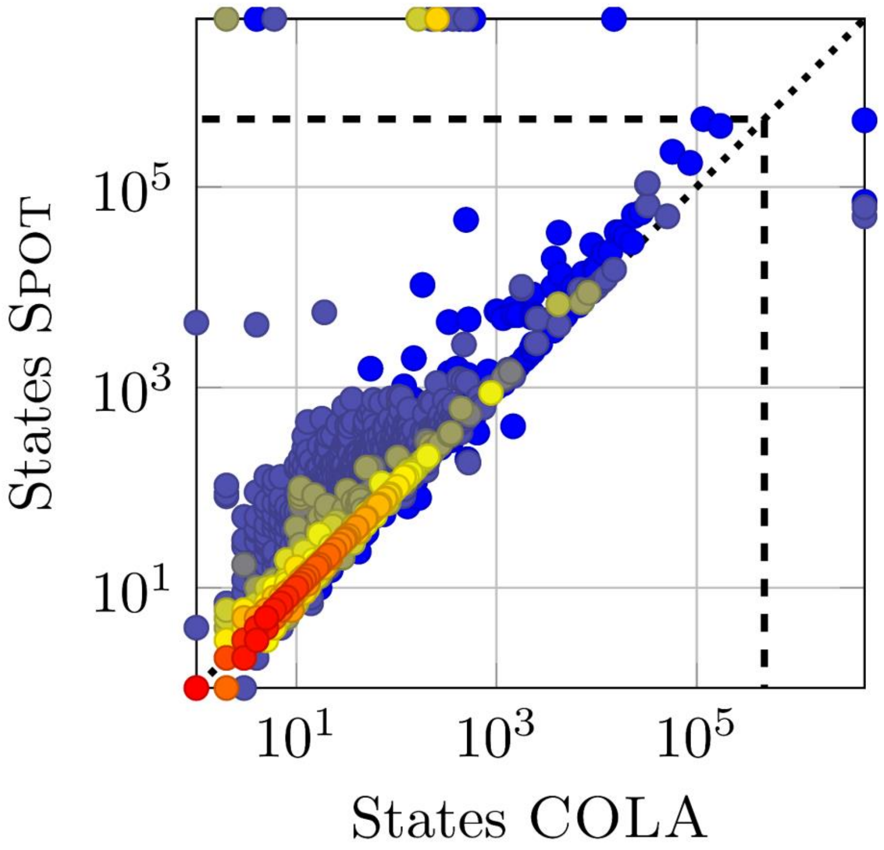
COLA solves **more instances** in **shorter** time



Tool	PAR-2 score: lower is better
COLA	**17,351**
Spot	67,258
Owl	206,431

Comparison with **Spot**

Heat map: blue color corresponds to fewer data points



COLA constructs

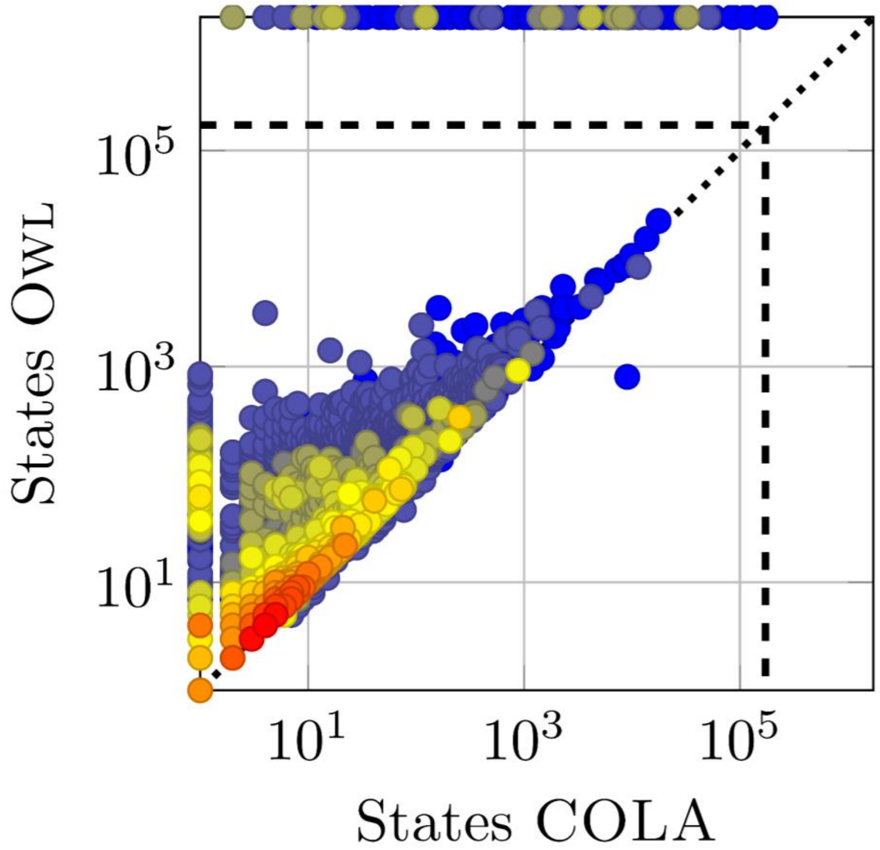
smaller

deterministic automata

than **Spot**

Comparison with **Owl**

Heat map: blue color corresponds to fewer data points



COLA constructs

smaller

deterministic automata

than **Owl**

Summary

1. **Divide-and-conquer** determinization
2. **Better upper bounds for two subclasses**:
   - $\mathbf{O}(n!)$ vs. $O((n!)^2)$ and $O(2^n)$ vs. $O(n!)$
3. **COLA** outperforms **Spot** and **Owl**

Future work

- **Parallel** determinization for each SCC
- Applications to
  - **Reactive synthesis**
  - **Probabilistic** verification
  - **Büchi complementation** and **inclusion**