ITMO UNIVERSITY

BFS-based Symmetry Breaking Predicates for DFA Identification



Vladimir Ulyantsev
PhD student
ITMO University

Ilya Zakirzyanov
Bachelor student
ITMO University

Anatoly Shalyto
Dr. Sci., professor
ITMO University

9th International Conference on Language and Automata Theory and Applications

March 4, 2015

ITMO UNIVERSITY

Presentation by

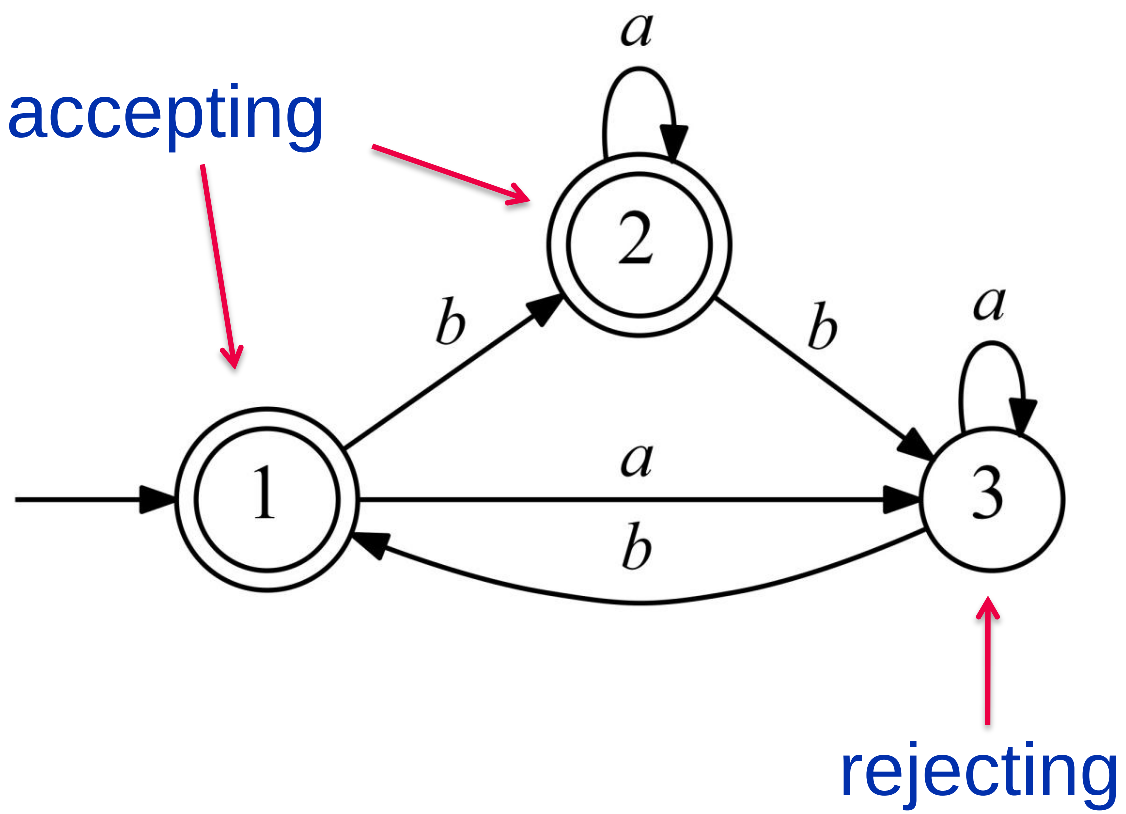
Daniil Chivilikhin
PhD student
ITMO University

ITMO UNIVERSITY

Outline

- Introduction
- DFASAT algorithm overview
- Handling noise in DFASAT
- BFS-based symmetry breaking for DFASAT
- Experiments
- Conclusions

Deterministic Finite Automata (DFA)

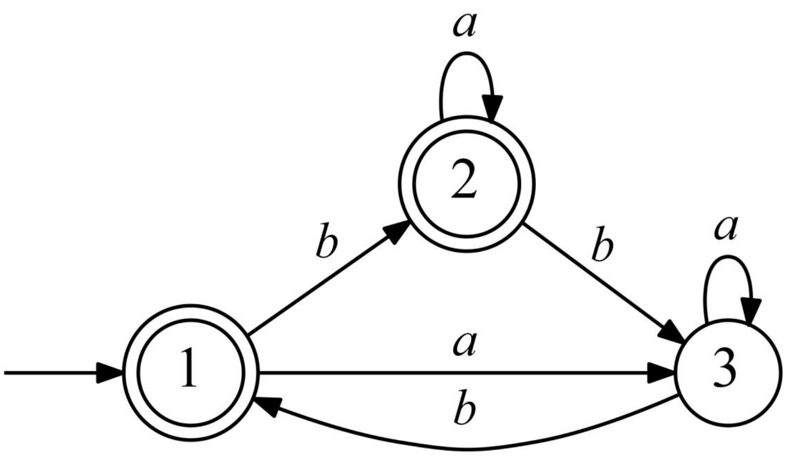


- S_+
  - ab
  - b
  - ba
  - bbb
- S_-
  - abbb
  - baba

DFA Identification Problem

S_+={ab, b, ba, bbb}

S_-={abbb, baba}



- Identifying a **minimal** DFA is NP-hard [Gold, 1978]

DFA Identification From Noisy Data

- K string labels are randomly flipped

S_+={ab, b, ba, bbb}; S_-={abbb, baba}

S_+={ab, b, ba}; S_-={abbb, baba, bbb}

Previous Research

- Evolutionary algorithm with smart state labeling [Lucas et al., 2005]
  - State of the art for noisy case
- **DFASAT [Heule & Verwer, 2010]**
  - **State of the art for noiseless case**

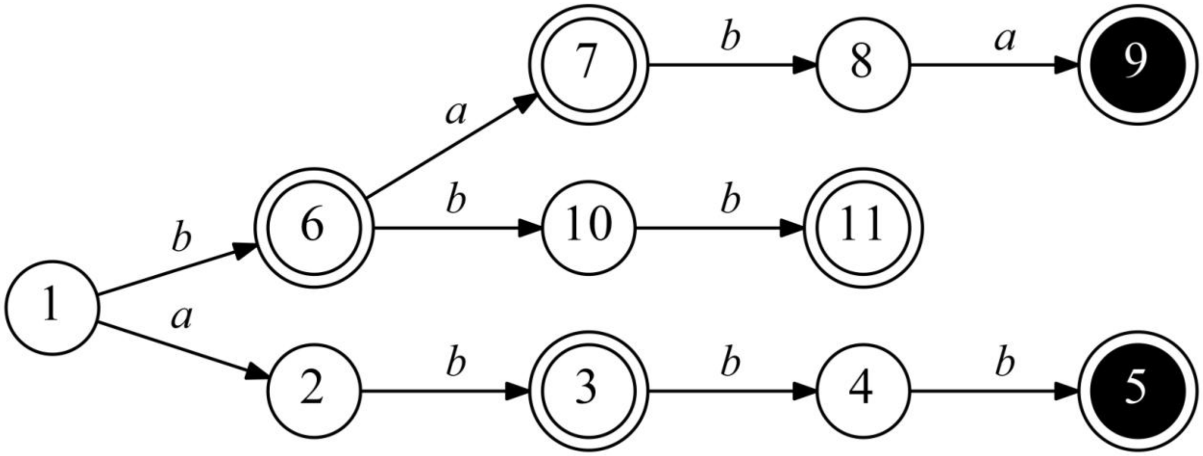
Our contribution

- We focus on DFASAT
- Augment DFASAT to handle **noisy data**
- Augment DFASAT with **new symmetry breaking predicates**

DFASAT [Heule & Verwer, 2010]

1. Augmented Prefix Tree Acceptor construction
2. Consistency Graph construction
3. CNF Boolean Formula construction
4. SAT-solver execution
5. DFA reconstruction from satisfying assignment

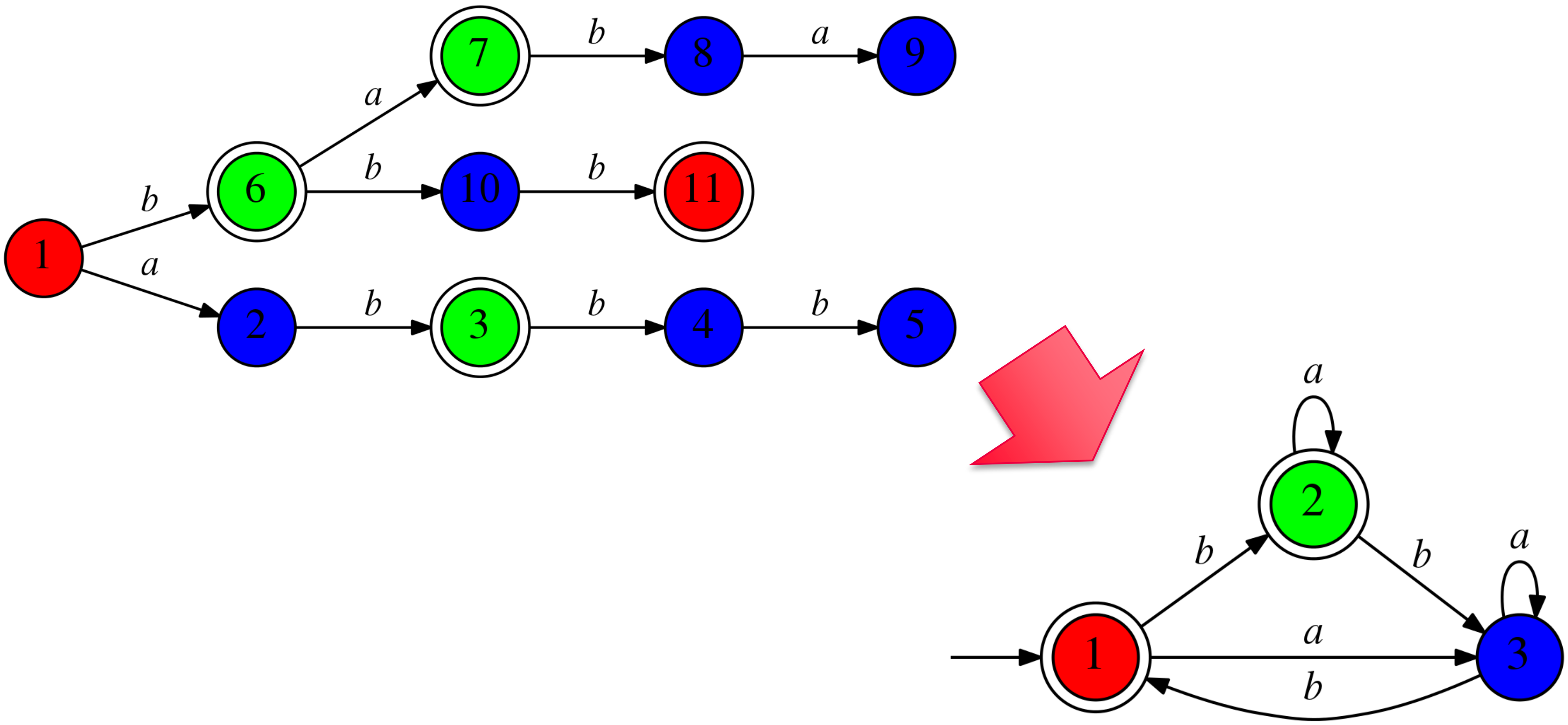
Augmented Prefix Tree Acceptor



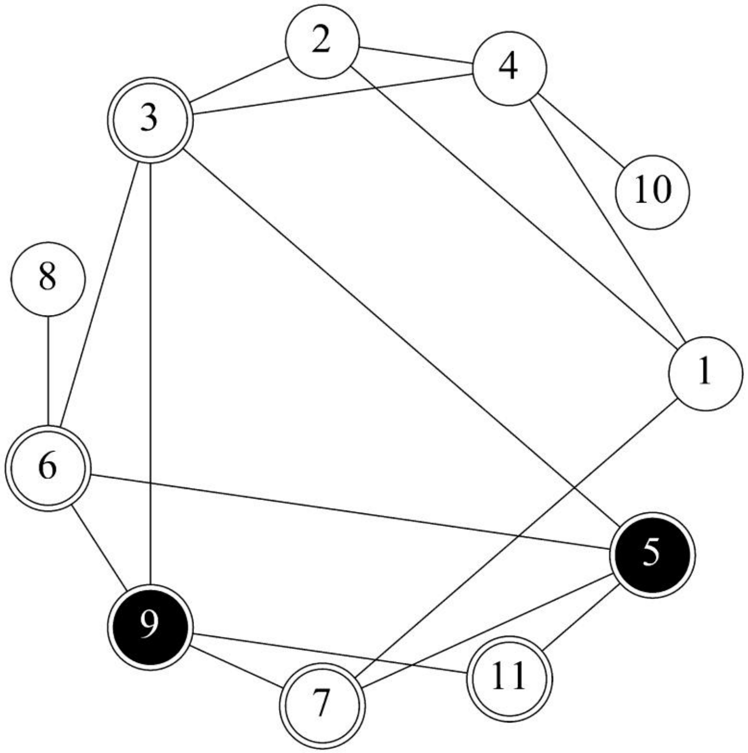
S_+
- ab
- b
- ba
- bbb

S_-
- abbb
- baba

Main idea: APTA coloring



Consistency Graph



- Nodes – same as APTA states
- Two nodes are connected if they cannot be merged into one DFA state
- Only exists in the noiseless case

Variables

- **Color** variables $x_{v,i} \equiv 1$ iff APTA state v has color i
- **Parent relation** variables $y_{l,i,j} \equiv 1$ iff DFA transition with symbol l from state i ends in state j
- **Accepting color** variables $z_i \equiv 1$ iff DFA state i is accepting

Types of clauses (1)

V_+ – accepting states
V_- – rejecting states

- **Accepting** states colors $x_{v,i} \Rightarrow z_i, \; v \in V_+$
- **Rejecting** states colors $x_{v,i} \Rightarrow \neg z_i, \; v \in V_-$
- Each state has **at least** one color $x_{v,1} \vee x_{v,2} \vee \ldots \vee x_{v,C}$
- Each state has **at most** one color $\neg x_{v,i} \vee \neg x_{v,j}, \; i < j$

Types of clauses (2)

p(*v*) – parent of APTA state v
l(*v*) – incoming symbol of APTA state *v*

- A DFA transition is set when a state and its parent are colored

$$x_{p(v),i} \wedge x_{v,j} \Rightarrow y_{l(v),i,j}$$

- Each DFA transition must target **at least** one state

$$y_{l,i,1} \vee y_{l,i,2} \vee \dots \vee y_{l,i,C}$$

- Each DFA transition can target **at most** one state

$$y_{l,i,j} \Rightarrow \neg y_{l,i,k}, \; j < k$$

Types of clauses (3)

- State color is set when DFA transition and parent color are set

$$y_{l(v),i,j} \wedge x_{p(v),i} \Rightarrow x_{v,j}$$

- Colors of two states connected with an edge in the consistency graph must be different

$$x_{v,i} \Rightarrow \neg x_{w,i}, (v,w) \in E$$

Noisy DFA Identification

- K random attribution labels are *flipped*

S_+={ab, b, ba, bbb}; S_-={abbb, baba}

S_+={ab, b, ba}; S_-={abbb, baba, bbb}

Noisy DFA Identification: Issues

- Consistency graph is undefined
- We do not know the exact labels of strings

- How can we modify the described translation to deal with noise?

Noisy DFA Identification (2)

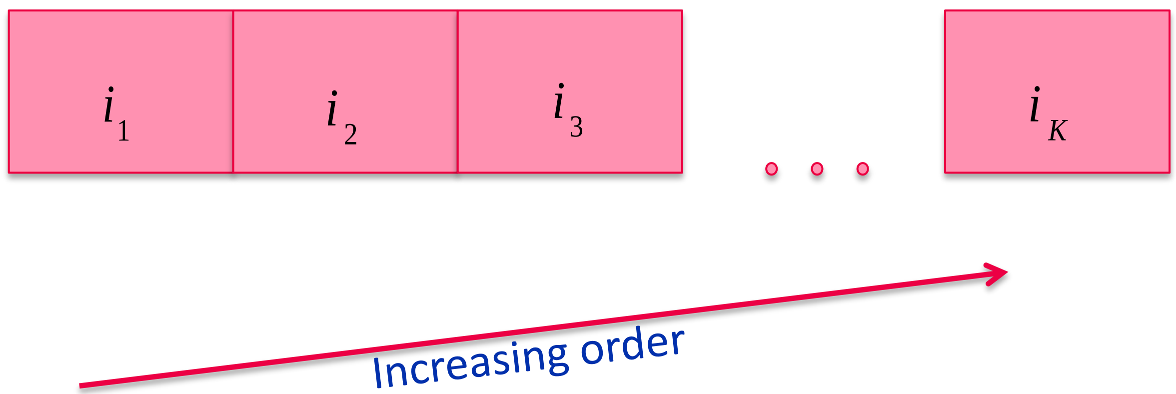
- New variables f_v
- $f_v \equiv 1$ iff the label of state v can (but <u>does not have to</u>) be incorrect (flipped)
- Modify clauses for state colors

$$x_{v,i} \Rightarrow z_i,\ v \in V_+ \qquad \longrightarrow \qquad \neg f_v \Rightarrow (x_{v,i} \Rightarrow z_i),\ v \in V_+$$

$$x_{v,i} \Rightarrow \neg z_i,\ v \in V_- \qquad \longrightarrow \qquad \neg f_v \Rightarrow (x_{v,i} \Rightarrow \neg z_i),\ v \in V_-$$

Noisy DFA Identification (3)

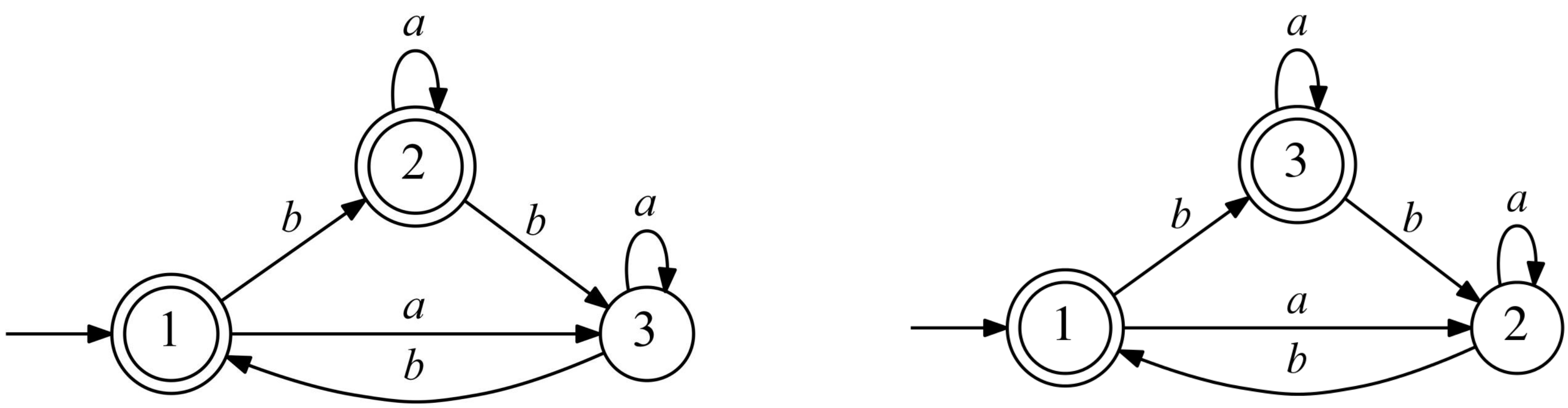
- Array of length *K*
- Numbers of APTA states for which that can be flipped



- Some extra variables and clauses for representing that as a Boolean formula; *order encoding* method used

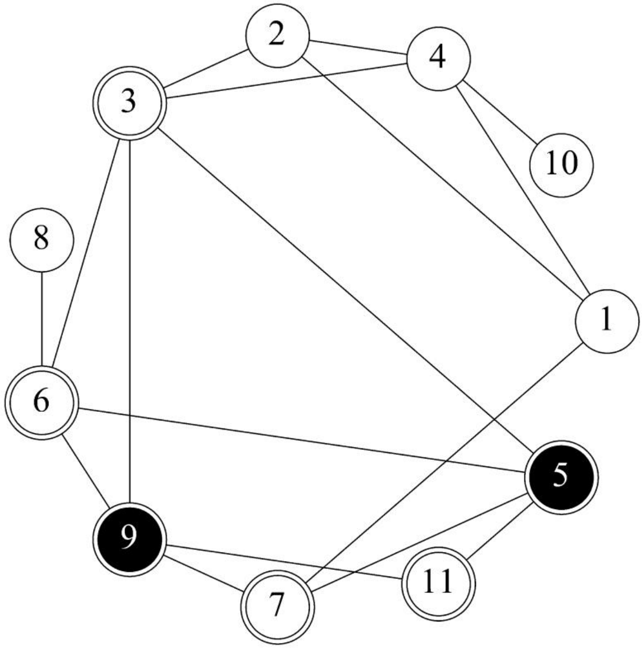
Symmetry breaking

- Many optimization problems exhibit symmetries
- Here: groups of isomorphic DFA

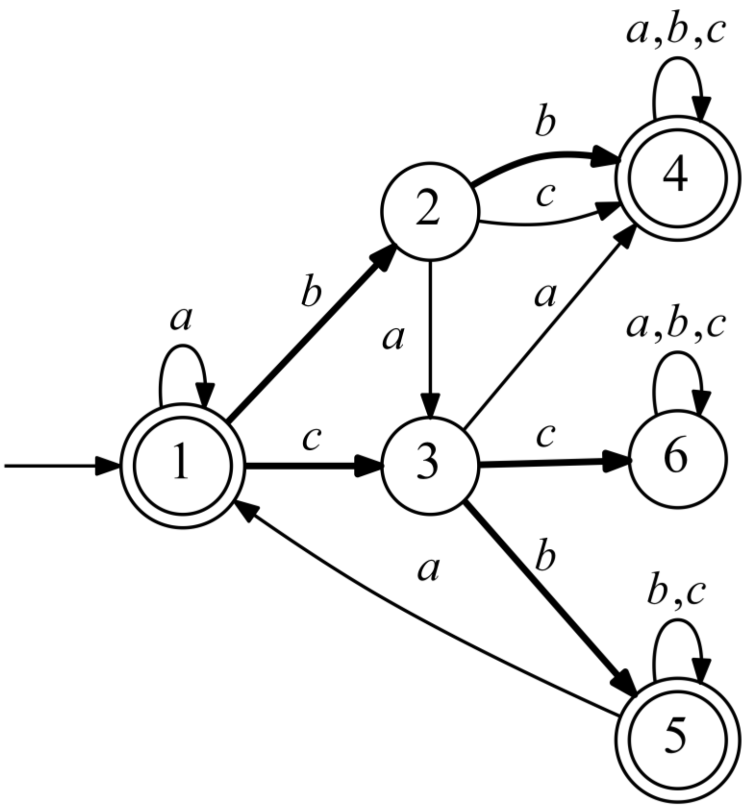


Max-clique symmetry breaking [Heule & Verwer, 2010]

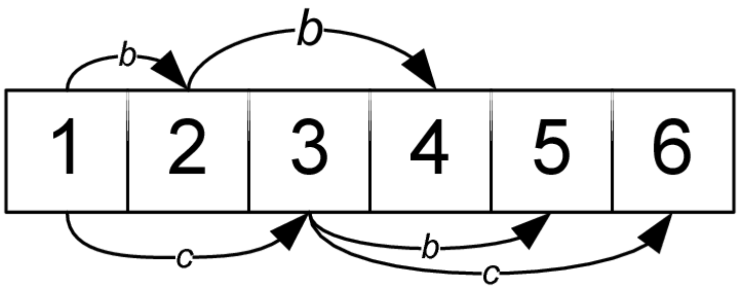
- Find a big clique in the CG with fast heuristic algorithm
- Fix colors of clique states in the APTA
- Note: not applicable in the noisy case



BFS-based Symmetry Breaking Predicates



BFS-enumerated DFA



BFS queue

BFS-based Symmetry Breaking Predicates

- Idea – force the DFA to be BFS-enumerated
- Already used in several algorithms

- How do we encode BFS-enumeration in SAT?

Additional variables

- **Parents** variables $p_{j,i} \equiv 1$ iff state i is the parent of state j in the BFS-tree

- **Transition** variables $t_{j,i} \equiv 1$ iff there is a transition between states i and j

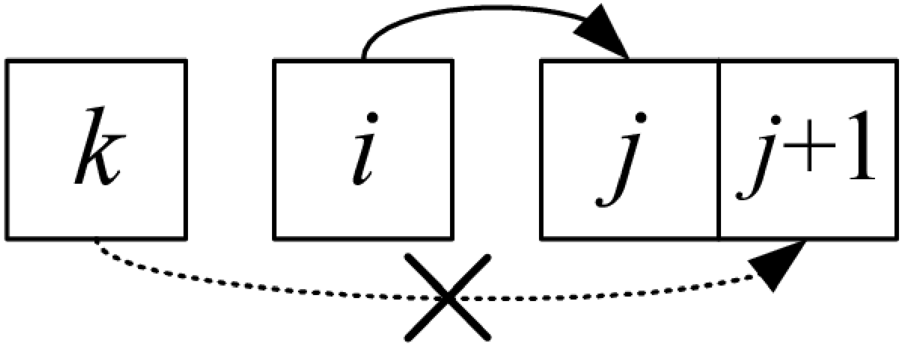
Ordering parents

- Each state except initial one must have a **parent** with a smaller number

$$p_{j,1} \vee p_{j,2} \vee \ldots \vee p_{j,j-1},\ 2 \le j \le C$$

- In BFS-enumeration states' **parents** must be ordered

$$p_{j,i} \Rightarrow \neg p_{j+1,k},\ 1 \le k < i < j < C$$



Ordering children

- Transition variables: there is a transition between states *i* and *j*

$$t_{i,j} \Leftrightarrow y_{l_1,i,j} \vee \ldots \vee y_{l_L,i,j},\ i < j$$

- State *j* was enqueued while processing the state with minimal number *i* among states that have a **transition** to *j*

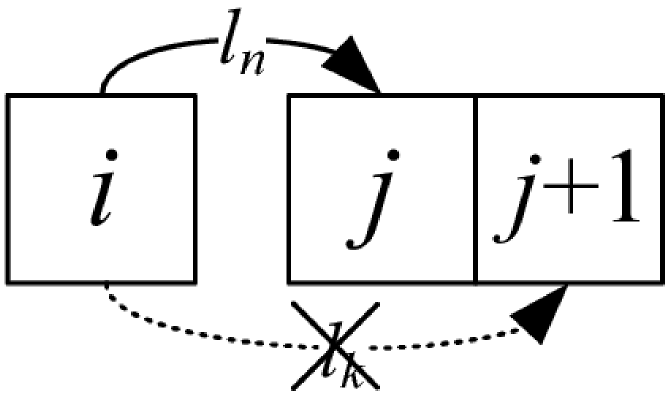
$$p_{j,i} \Leftrightarrow (t_{i,j} \wedge \neg t_{i-1,j} \wedge \ldots \wedge \neg t_{1,j}),\ i < j$$

Ordering transitions

- Minimal symbol variables

$$m_{l_n,i,j} \Leftrightarrow y_{l_n,i,j} \wedge \neg y_{l_{n-1},i,j} \wedge \ldots \wedge \neg y_{l_1,i,j},\ i < j$$

- Arranging consecutive states *j* and *j+1* with the same **parent** *i* in the alphabetical order of **minimal symbols** on **transitions** between them and *i*



$$p_{j,i} \wedge p_{j+1,i} \wedge m_{l_n,i,j} \Rightarrow \neg m_{l_k,i,j+1},\ i < j,\ k < n$$

Experimental setup

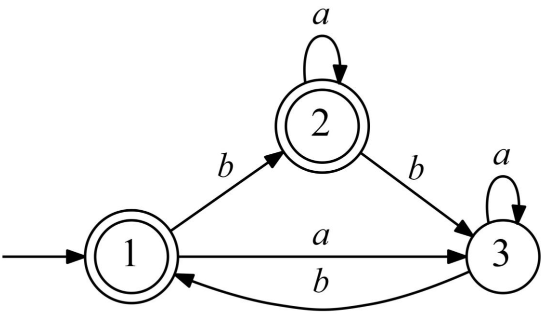
- Random data sets
- Binary alphabet
- *TL* – time limit (*TL* = 1800 seconds)
- *lingeling* SAT-solver
- Mean time among 100 launches of experiments

Noiseless DFA Identification

- DFASAT with max-clique symmetry breaking clearly outperforms our method

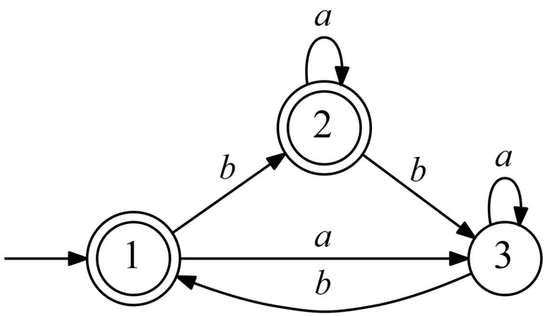
Noisy DFA Identification when target DFA exists

- *N* – size of the DFA used for generating input set of strings
- *N* – size of the target DFA



N states

S_+={ab, b, ba, bbb}
S_-={abbb, baba}



N states

Noisy DFA Identification, $S = 10N$ strings

Number of states	Noise level, %	BFS, s	DFASAT, s	EA, s
5	2	**0.22**	0.38	1.22
5	4	**0.59**	0.9	1.1
6	2	**1.05**	2.44	2.94
6	4	3.34	7.82	**2.85**
7	1	**4.34**	10.83	21.36
7	3	**17.22**	143.66	19.16
8	1	**17.89**	31.58	30.29
8	2	163.92	225.31	**19.8**

Noisy DFA Identification, *S* = 25*N* strings

Number of states	Noise level, %	BFS, s	DFASAT, s	EA, s
5	1	**0.54**	0.64	2.77
5	2	2.42	4.33	**1.80**
6	1	**6.3**	11.95	11.65
6	2	13.3	43.54	**4.8**
7	1	31.01	114.95	**17.24**
7	2	286.76	TL	**13.11**
8	1	239.46	404.32	**21.73**

Noisy DFA Identification, *S* = 50*N* strings

Number of states	Noise level, %	BFS, s	DFASAT, sec	EA, s
5	1	**4.2**	7.59	6.07
5	2	12.87	22.36	**3.05**
6	1	20.76	52.5	**20.39**
6	2	107.94	309.22	**11.28**

Noisy DFA identification when the target DFA does not exist

- (N + 1) – size of the DFA used for generating input set of strings
- N – size of the target DFA

- Note: the state-of-the-art EA cannot determine that a DFA consistent with a given set of strings **does not exist**

Noisy DFA identification when the target DFA does not exist, $S = 50N$ strings

N	K	BFS, s	DFASAT, s	Passed BFS, %	Passed DFASAT, %
5	1	**11.57**	257.13	100	100
5	2	**46.42**	1296.71	100	30
6	1	**110.05**	TL	100	0
6	2	**581.73**	TL	100	0
7	1	**995.27**	TL	89	0
7	2	TL	TL	0	0

Conclusion

- Exact solution for noisy DFA identification
- New symmetry breaking predicates based on BFS
  - Applicable in the noisy case
  - Greatly speed up the discovery of non-existence of a DFA
- Implementation
  - http://github.com/ctlab/DFA-Inductor

Acknowledgements

- This work was financially supported by the Government of Russian Federation, Grant 074-U01, and also partially supported by RFBR, research project No. 14-07-31337 mol_a.

ITMO UNIVERSITY

Thank you for your attention!

Vladimir Ulyantsev
Ilya Zakirzyanov
Anatoly Shalyto
{ulyantsev,zakirzyanov}@rain.ifmo.ru