

Generation of DFA Minimization Problems

Gregor Sönnichsen

Universität Bayreuth

21. Mai 2020

Automata theory is a classical topic in computer science curricula.
Minimization of DFAs is a typical task for students:

- ▶ sufficiently easy to understand
- ▶ practical applications
- ▶ understanding can be tested easily

Consequently, studying automatized generation of DFA minimization problems is interesting because it could...

- ▶ ... free up precious time
(if a generator is implemented)
- ▶ ... yield a deeper insight
in the nature of such problems



Outline

Introduction

Problem definition and approach

Introduction

Preliminaries (1/2)

A tuple $A = (Q, \Sigma, \delta, s, F)$ with Q, Σ being a finite, $\delta: Q \times \Sigma \rightarrow Q$, $s \in Q$ and $F \subseteq Q$ is called *deterministic finite automaton*.

We define the *extended transition function* $\delta^*: Q \times \Sigma^* \rightarrow Q$ as:

- ▶ $\delta^*(q, \varepsilon) = q$
- ▶ $\delta^*(q, w\sigma) = \delta(\delta^*(q, w), \sigma)$ for all $q \in Q$, $w \in \Sigma^*$, $\sigma \in \Sigma$

The *language* of DFA is defined as $L(A) = \{ w \mid \delta^*(w) \in F \}$.

We call a DFA *minimal*, if there exists no other DFA with the same language having less states.

Introduction

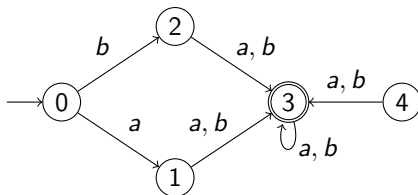
Preliminaries (2/2)

We say a state q is *unreachable*, iff there is no word $w \in \Sigma^*$ such that $\delta^*(s, w) = q$.

A state pair $q_1, q_2 \in Q$ is called *equivalent*, iff $\sim_A(q_1, q_2)$ is true, where

$$q_1 \sim_A q_2 \Leftrightarrow_{\text{def}} \forall z \in \Sigma^*: (\delta^*(q_1, z) \in F \Leftrightarrow \delta^*(q_2, z) \in F)$$

Example



Theorem

A DFA is minimal, iff it has neither unreachable nor equivalent states.

Hopcroft's Minimization Algorithm

1. Compute all unreachable states
2. Remove all unreachable states and their transitions
3. Compute all equivalent state pairs (\sim_A)

```
1: function FindEquivPairs( $A$ )
```

2: $i \leftarrow 0$

3: $m(0) \leftarrow \{(p, q), (q, p) \mid p \in F, q \notin F\}$

4: **do**

5: $i \leftarrow i + 1$

$$6: \quad m(i) \leftarrow \{(p, q), (q, p) \mid (p, q) \notin \bigcup m(\cdot) \wedge$$

7: $\exists \sigma \in \Sigma: (\delta(p, \sigma), \delta(q, \sigma)) \in m(i-1)\}$

```
8:   while  $m(i) \neq \emptyset$ 
```

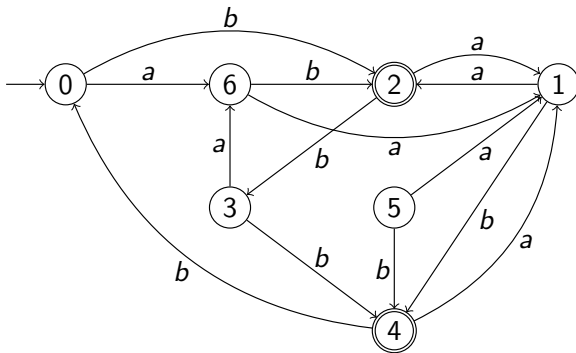
```
9:   return  $\bigcup m(\cdot)$ 
```

4. Merge all equivalent state pairs

Introduction

A sample DFA minimization problem...

Task: Consider the below shown deterministic finite automaton A :



Apply the minimization algorithm and illustrate for each state pair of A during which FindEquivPairs-iteration it was marked. Draw the resulting automaton.

Introduction

... and its solution

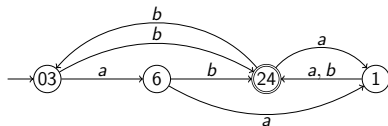
Solution:

Step 1: Detect and eliminate unreachable states.

State 5 is unreachable.

Step 2: Apply FindEquivPairs to A and merge equivalent state pairs:

	0	1	2	3	4	6
0	■	1	0		0	2
1	■	■	0	1	0	1
2	■	■	■	0		0
3	■	■	■	■	0	2
4	■	■	■	■	■	0
6	■	■	■	■	■	■



Conclusion

This presentation has...



References