Generation of DFA Minimization Problems

Gregor Sönnichsen

Universität Bayreuth

22. Mai 2020

Introduction

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

[1]

[1] https://www.rindlerwahn.de/zeitdiebe-besiegen-und-mehr-lebenszeit-gewinnen

Outline

Problem definition

Preliminaries (1/2)

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

We define the *extended transition function* $\delta^* : Q \times \Sigma^* \to Q$ 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.

Problem definition

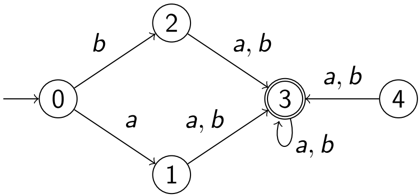
Preliminaries (2/2)

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

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

Example



Theorem

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

Problem definition

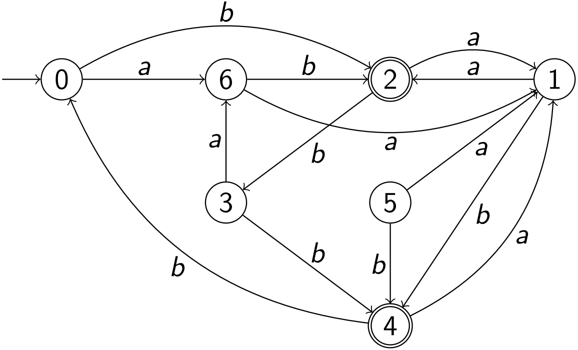
Hopcroft's Minimization Algorithm

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

    function FindEquivPairs(A)
    $\quad i \leftarrow 0$
    $\quad m(0) \leftarrow \{(p, q), (q, p) \mid p \in F, q \notin F\}$
    $\quad$ **do**
    $\quad\quad i \leftarrow i + 1$
    $\quad\quad m(i) \leftarrow \{(p, q), (q, p) \mid (p, q) \notin \bigcup m(\cdot) \wedge$
    $\quad\quad\quad\quad \exists \sigma \in \Sigma\colon (\delta(p, \sigma), \delta(q, \sigma)) \in m(i - 1)\}$
    $\quad$ **while** $m(i) \neq \emptyset$
    $\quad$ **return** $\bigcup m(\cdot)$

4. Merge all equivalent state pairs

Task: *Consider the below shown deterministic finite automaton A:*



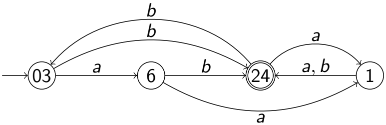
Apply the minimization algorithm and illustrate for each state pair of A during which FindEquivPairs-*iteration it was marked. Draw the resulting automaton.*

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	■	■	■	■	■	■



Approach

Generating Minimal DFAs

Approach

Generating Minimal DFAs

Test DFAs

Duplicating a State

Extending Minimal DFAs

Adding Equivalent States (2)

Addding outgoing transitions

Extending Minimal DFAs

Adding Equivalent States (3)

Adddding ingoing transitions

Algorithm

Extending Minimal DFAs

Adding Unreachable States

Conclusion

This presentation has. . .

-

References