# Generation of DFA Minimization Problems

Gregor Sönnichsen

Universität Bayreuth

31. 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 for exercise constructors (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 $Q, \Sigma$ 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

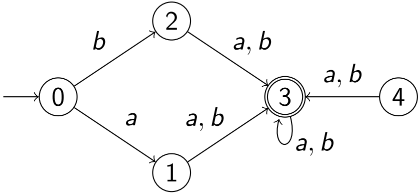
## Preliminaries (2/2)

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

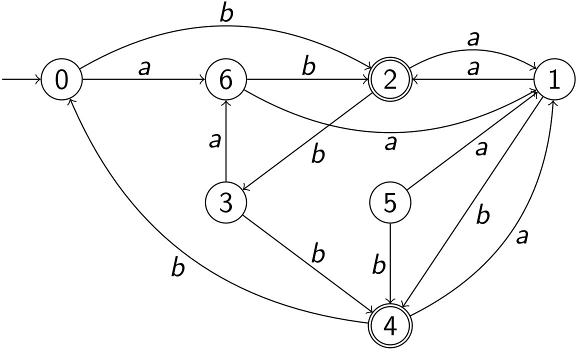
## Hopcroft's Minimization Algorithm

1. Compute all unreachable states
2. Remove all unreachable states and their transitions
3. Compute all inequivalent state pairs ($\not\sim_A$)

```
function FindEquivPairs(A)
    i ← 0
    m(0) ← {(p, q), (q, p) | p ∈ F, q ∉ F}
    do
        i ← i + 1
        m(i) ← {(p, q), (q, p) | (p, q) ∉ ⋃ m(·) ∧
                                   ∃σ ∈ Σ: (δ(p, σ), δ(q, σ)) ∈ m(i − 1)}
    while m(i) ≠ ∅
    return ⋃ m(·)
```

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.*

# Problem definition

## . . . and its solution

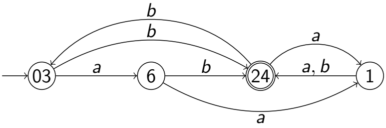
*Solution:*

*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 | ■ | ■ | ■ | ■ | ■ | ■ |



# Approach

"Reversing Hopcroft's Algorithm" Modular Pipeline-Architecture

# Outline

# Generating Minimal DFAs

## Rejection Algorithm

Approach: Generate test DFAs until they match the demanded properties.

**function** GenNewMinDFA $(n_s, k, n_F, d, p)$
    $I \leftarrow$ all DFAs in $\mathrm{DB}_{found}$ matching $n_s, k, n_F$
    **while** True **do**
        generate DFA $A_{test}$ with $|Q|, |\Sigma|, |F|$ matching $n_s, k, n_F$
        **if** $A_{test}$ not minimal **or** $d \neq \mathfrak{D}(A_{test})$ **then**
            **continue**
        **if** $p = 1$ **and** $A_{test}$ is not planar **then**
            **continue**
        **if** $A_{test}$ is isomorph to any DFA in $I$ **then**
            **continue**
        save $A_{test}$ and its respective properties in $\mathrm{DB}_{found}$
        **return** $A_{test}$

We will restrict ourselves to the following DFAs:

$$Q = \mathbb{N}_0^{n_s - 1}, \Sigma = \mathbb{N}_0^{k-1}, s = 0$$

Generation by Randomization:

$$\begin{aligned} F &= random\_subset(Q) \\ \forall q \in Q, \sigma \in \Sigma \colon \delta(q, \sigma) &= choose\_one(Q) \end{aligned}$$

Generation by Enumeration:

An *enumeration state* $s_{n_s,k,n_F} = (F_F, F_\delta)$ consists of two arrays of length $n_s, n_s * k$, respectively.

$$F_F[i] = 1 \Leftrightarrow_{def} i \in F$$
$$F_\delta[i * k + j] = q \Leftrightarrow_{def} \delta(i, j) = q$$

Short demo how the associated DFA changes, when an enumeration state is incremented

# Outline

Let the added states be $r_1, \ldots, r_{n_e}$, such that $Q_{re} = Q_{sol} \cup \{r_1, \ldots, r_{n_e}\}$ and each of the added states is equivalent to a distinct state in $Q_{re}$.

Observation: Each $r_i$ will be equivalent to a state $e$ of $A_{sol}$:

$$\forall i \in [1, n_e]: \;\; \exists e \in Q_{sol}: \;\; r_i \sim_A e$$

We will first choose an $e \in Q_{sol}$ for each state $r_i$ we create, then we add its transitions.

Outgoing transitions. Observation:

$$
\begin{aligned}
& r_i \sim_A e \\
\Rightarrow\ & \forall \sigma \in \Sigma \colon [\delta(r_i, \sigma)]_{\sim_A} = [\delta(e, \sigma)]_{\sim_A}
\end{aligned}
$$

We see that $\delta(r_i, \sigma) = q$ must be in the same equivalency class as $\delta(e, \sigma) = p$.

Consequently:

R1: For each symbol $\sigma \in \Sigma$ choose exactly one state $q \in [\delta(e, \sigma)]_{\sim_A}$ and set $\delta(r_i, \sigma) = q$.

The rule is always fulfillable:

- $A_{sol}$ is complete
- every $r_i$ gets an out. transition for every alphabet symbol

# Extending Minimal DFAs

## Adding Equivalent States (3)

Ingoing transitions. Observations:

- $r_i$ must be reachable $\Rightarrow$ $in(r_i) >= 1$

Let $q$ be a state that gets an in. transition to $r_i$.

$q$ must remain in its equivalence class

$\Rightarrow$ $q$ has to have a transition to some state in $[r_i]_{\sim_A} = [e]_{\sim_A}$

$\Rightarrow$ given $\delta(q, \sigma) = p$ and $p \in [e]_{\sim_A}$, we can set $\delta(q, \sigma) = r_i$

We see that $p$ must have at least 2 ingoing transitions.

R2: Choose at least one $((q, \sigma), p) \in \delta$ with $[p] = [e]$ and $out(p) \geq 2$. Remove $((q, \sigma), p)$ from $\delta$ and add $((q, \sigma), r_i)$.

General requirement regarding the choice of a state $e$ for an $r_i$:

$$duplicatable(q) \Leftrightarrow_{def} (\exists p \in [q]_{\sim_A} : out(p) \geq 2)$$

Example

# Extending Minimal DFAs

## Adding Unreachable States

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

**function** AddUnrStates $(A, n_u, c)$
  $U \leftarrow \emptyset$
  **for** $n_u$ **times do**
    let $q$ be the new state
    add ingoing tr. from a random subset of $U \times \Sigma$
    $\Sigma' \leftarrow$ **if** $c = 1$ **then** $\Sigma$ **else** random subset of $\Sigma$
    add outgoing tr. to $|\Sigma'|$ random states
    add $q$ to $U$
  **return** $A$

# Outline

# Live Demonstration

## Module overview

# Live Demonstration

## Command-Line Options

# Conclusion

This work has...

- 

Lookout...

- 

# References